НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки”

на тему: Інформаційна технологія для ефективного менеджменту хронічних захворювань

Виконав: студент 4 курсу, групи ТР-52

Басалик Гліб Андрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викл. Бандурка О.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Басалику Глібу Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Інформаційна система для ефективного менеджменту хронічних захворювань”

керівник роботи ст.викл. Бандурка Олена Іванівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи персональний комп’ютер під керуванням операційної системи Microsoft Windows, мова програмування C#.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та можливі засоби реалізації взаємодії, обґрунтувати обрані програмні застосунки та шляхи розробки програмних додатків, розробити програмне забезпечення, розробити користувацький інтерфейс, зробити висновки за результатами роботи
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень) 1. Актуальність завдання. 2. Мета розробки 3. Функції системи моделювання. 4. Методи розв’язку задачі. 5. Індивідуально-орієнтована модель системи. 6. Марковський процес прийняття рішень. 7. Схема роботи програмного додатку. 8. Інтерфейс користувача. 9. Висновки.

6. Публікації: _____

Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Басалик Г. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бандурка О. І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи було створення інформаційної системи, що вирішує задачу ефективного менеджменту хронічних вірусних захворювань. Система генерує оптимальний варіант стратегії діагностик та вакцинацій хронічних захворювань, прогнозує розвиток популяції людей, враховуючи прогресію хвороби. Програма забезпечує можливість генерувати як повну оптимальну стратегію, так і окремі її напрями, графіки, які показують відношення контрольної популяції та оптимальної, а також загальні графіки залежностей окремих параметрів. Користувацький додаток представляє собою демонстрацію роботи розробленого алгоритму.

Ключові слова: C#, ХРОНІЧНЕ ЗАХВОРЮВАННЯ, РШМ, ДИСПЛАЗІЯ, ЕФЕКТИВНИЙ МЕНЕДЖМЕНТ, ІНФОРМАЦІЙНА СИСТЕМА, ВПЧ.

Записка містить 82 сторінки, 25 рисунків та 25 посилань.

ABSTRACT

The purpose of the work was an information system that solves the problem of effective management of chronic viral diseases. The system generates an optimal version of the strategy of diagnosis and vaccination of chronic diseases, predicts the development of the population of people, taking into account the progression of the disease. The program provides the opportunity to generate both a complete optimal strategy, as well as its individual directions, graphs showing the relation of the control population and the optimal, as well as general graphs of the dependencies of individual parameters. The custom application is a demonstration of the work of the developed algorithm.

Keywords: C#, CHRONIC DISEASES, CCHM, DISPPLAIN, EFFICIENT MANAGEMENT, INFORMATION SYSTEM, HPV.

The note contains 82 pages, 25 images and 25 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE (англ. Integrated development environment) – інтегроване середовище розробки;

API (англ. Application Programming Interface) – прикладний програмний інтерфейс;

РШМ – рак шийки матки;

Cin I – дисплазія початкового рівня складності;

Cin II-III – дисплазія середнього або важкого рівня складності;

ВПЛ – вірус папіломи людини;

Скринінг – стратегія в організації охорони здоров'я, спрямована на виявлення захворювань у клінічно безсимптомних осіб в популяції.

ВСТУП

Належне використання етіологічної діагностики має важливе значення для отримання позитивних результатів фармакотерапії та вдосконалення системи охорони здоров'я, що показали інновації останніх років в діагностиці хронічних вірусних інфекцій.

Як невід'ємний компонент якісної фармацевтичної допомоги, діагностика вірусних інфекцій, особливо хронічних, є основою прийняття багатьох рішень, надаючи ключову інформацію, необхідну для специфічної профілактики та фармакотерапії вірусного захворювання [1].

Метою діагностичного скринінгу є раннє виявлення та точний діагноз хронічного вірусного захворювання, ранній початок та належна фармакотерапія виявленої вірусної інфекції, менша кількість рецидивів або гострих епізодів, більш повільне прогресування вірусного захворювання та менші втрати працездатності.

Оцінка ефективності етіологічної діагностики хронічних вірусних інфекцій є ключовою для розробки та реалізації оптимальних стратегій боротьби з ними для окремих груп населення, розробки рекомендацій щодо належного використання діагностичних тестів, визначення оптимального рівня охоплення діагностичним скринінгом та визначення його ефективності.

Вчасне виявлення етіологічного агенту сприяє виявленню вірусних інфекцій на ранніх стадіях до появи клінічних проявів, вибору безпечних та ефективних методів фармакотерапії, плануванню стратегій боротьби з вірусними захворюваннями, оцінці ефективності всіх етапів надання медичної та фармацевтичної допомоги [2].

Вирішення питання ефективності менеджменту аналітичним шляхом – практично неможливе, через велику кількість впливаючих факторів. А існуючого програмного забезпечення для вирішення конкретної проблеми ефективності

менеджменту хронічних інфекційних захворювань та пошуку оптимальної стратегії фактично не існує.

Проведене дослідження сприяло розробці індивідуально-орієнтовної моделі та вибору алгоритму її навчання для прогнозування поширення і розвитку хронічних вірусних інфекцій серед певних популяцій людей, а також дослідження ефективності стратегії проведення діагностичного скринінгу та фармацевтичних втручань і їх оптимізації.

Програмний додаток був реалізований за допомогою об'єктно-орієнтовної мови програмування C#. Для створення інформаційної системи були використані інтерфейс програмування Windows Forms, для побудови візуально привабливої та зручної для користувача системи, а також додаткові об'єкти бібліотеки Microsoft.Office.Interop.Excel, які надають змогу зчитувати та записувати інформацію в документах Microsoft Office Excel.

Під час розробки програмного продукту були використані метод Монте-Карло та марковський процес прийняття рішень. В основі методу Монте-Карло лежить алгоритм генерації випадкової вибірки для вирішення проблем. Він був обраний, як найбільш зручний для задач оптимізації.

Марківські ланцюги, було використано як найбільш оптимальний для опису та аналізу послідовних рішень в умовах невизначеності. Динамічне програмування допомагає визначити оптимальну стратегію прийняття рішення і більш адекватну модель поведінки реальних осіб, які приймають рішення.

Розроблений додаток під час моделювання системи враховує такі параметри, як розмір популяції, яка буде змодельована, значення вхідної групи людей та їх розподіл за віком, станом хвороби і групою вірусу папіломи людини. Програмний продукт забезпечує можливість генерувати як повну оптимальну стратегію, так і окремі її напрями, графіки, які показують відношення контрольної популяції та оптимальної, а також загальні графіки залежностей окремих параметрів. Користувацький додаток представляє собою демонстрацію роботи розробленого алгоритму, виведення графіків, даних про популяції та стратегії фармакотерапевтичних втручань і генерацію звіту.

Записка містить 5 розділів.

У першому розділі описується постановка задачі інформаційної технології для ефективного менеджменту хронічних захворювань.

У другому розділі описуються підходи до вирішення проблеми ефективного менеджменту захворювань та їх аналіз.

У третьому розділі вказуються основні засоби розробки даної системи.

У четвертому розділі дано опис реалізованого програмного продукту та його архітектури.

У п’ятому розділі описано роботу користувача з програмною системою.

1 ПОСТАНОВКА ЗАДАЧІ ЕФЕКТИВНОГО МЕНЕДЖМЕНТУ ХРОНІЧНИХ ЗАХВОРЮВАНЬ

Завданням роботи була реалізація інформаційної системи для ефективного менеджменту хронічних інфекційних захворювань, за допомогою якої можна проаналізувати розвиток захворювання серед певної популяції та його наслідки [3]. Система повинна запропонувати оптимальну стратегію боротьби з цим захворюванням, що відбувається за рахунок діагностик та вакцинацій населення.

В результаті розробки програмного продукту, спеціалісту з медичної аналітики буде надана можливість опрацьовувати дані різних популяцій людей, з урахуванням розвитку хронічного інфекційного захворювання, прогнозувати їх подальші перетворення і отримувати найбільш ефективні стратегії втручання в розвиток цього захворювання у популяції. За рахунок цього спеціаліст зможе попередити більшу кількість летальних випадків та загострень захворювання, ніж це можливо зараз.

Розроблена система повинна забезпечувати наступні можливості:

– зчитування вхідних даних з Excel документа;

– масштабування контрольної та оптимальної популяції людей перед їх моделюванням;

– калібрування системи відносно вхідної групи людей, що дасть змогу знайти матриці переходів від одного стану хвороби до іншого та від однієї групи ВПЛ до іншого;

– пошук імовірностей переходів між станами захворювання, а також між групами ВПЛ та виведення відповідних значень на екран користувача в зрозумілому форматі;

– моделювання певної контрольної популяції, в розвиток захворювання якої не втручаються (тобто вірогідність фармакотерапевтичних заходів для цієї популяції та витрати на них рівні нулю);

– отримання оптимальних показників щодо імовірності проходження діагностик та вакцинацій, а також розподілу бюджету серед них;

– моделювання популяції, в якій втручання в розвиток хронічного захворювання проходить за оптимальною стратегією набору діагностичних заходів та вакцинацій;

– регулювання кожного зі стратегічних факторів окремо та моделювання популяції людей за відповідними імовірностями фармакотерапевтичних заходів відносно цієї популяції та затратами на них;

– вивід та збереження графіків залежності кількості людей у різних станах від стратегічних факторів;

– просте та зрозуміле порівняння контрольної та оптимальної популяції у вигляді графічного представлення;

– порівняння ефективності медичного втручання в оптимальну популяцію та контрольну;

– збереження графіків залежності між певними заходами, які будуються з набору точок, які випадково моделює система;

– збереження звіту з графіками та тестами різних варіантів стратегій у форматі .xls.

Для розробки інформаційної системи була використана мова програмування C#. Розробка графічного інтерфейсу користувача відбувалась на основі Windows Forms.

Метою розробки інформаційної системи менеджменту хронічних вірусних захворювань є створення програмного продукту, що дасть змогу отримати дані, спираючись на які, медичні центри та лікарні зможуть більш ефективно боротися із захворюваннями. Це дасть зважено та чітко оцінювати небезпеку поширення інфекції серед груп людей та правильно розставляти акценти на діагностиках та вакцинаціях у стратегіях боротьби проти захворювань. Також за допомогою цієї системи будуть чітко розставлені пріоритети у розподілі бюджету між заходами, направленими на боротьбу із хронічними захворюваннями [4].

Завдання проекту – надання змоги аналітику у медичній сфері з будь-яким досвідом користування комп’ютером здійснити аналіз вибірки людей, яка є у нього, щодо розвитку у неї хронічних захворювань, при цьому потребуючи мінімум інформації про конкретну популяцію. Цей проект дає змогу аналітикам дізнатися найефективнішу стратегію діагностики та вакцинації для даної йому популяції. А також об’єктивно та наглядно порівняти запропоновану системою стратегію медичного втручання та існуючу в даний момент [5].

2 АНАЛІЗ ПРОБЛЕМИ ЕФЕКТИВНОГО МЕНЕДЖМЕНТУ ХРОНІЧНИХ ЗАХВОРЮВАНЬ

В більшості медичних центрів існують інформаційні бази, що містять в собі дані про пацієнтів. Вони досить громіздкі та містять велику кількість інформації по цілим групам людей. Серед цих даних також є відомості про хронічні захворювання кожної людини, а отже й конкретної популяції. Ці бази даних формують і зберігають лікарні та медичні центри. Потрібну інформацію від них можуть отримати різні інститути та заклади, які займаються прогнозуванням розвитку певних захворювань.

В зв'язку зі складною предметною областю, рішень, які б повністю відповідали завданню, приведеному вище, просто не існує. Більшість із запропонованих в даний момент інформаційних систем охоплюють інші хвороби та не мають можливості бути адаптованими саме для хронічних захворювань. Частина систем менеджменту захворювань призначена для вирішення інших завдань. Також переважна частина цих систем призначені для комплексного введення в експлуатацію в усі медичні заклади та використання, що практично неможливо за умов недостатнього фінансування та відсутності обладнання і достатньої кількості кваліфікованих ІТ-спеціалістів в наших медичних закладах [6].

Але для аналітики та звітування в медичній сфері часто використовують пакет програм MS Office, MatLab та MathCad. Його достатньо для базових підрахунків та їх представлення у вигляді звітів з графічним відображенням.

Програмних продуктів, здатних змоделювати популяцію з урахуванням хвороби та втручання в її розвиток, в наш час у вільному доступі взагалі немає. А це лишається основною задачею перед інформаційною системою, що була описана вище.

Моделювання популяцій та прогнозування їх подальшого розвитку в запропонованому програмному продукті спираються на принципи індивідуально-орієнтовного моделювання [7].

2.1 Програмні продукти MS Office

Пакет Microsoft Office був створений корпорацією Microsoft та доступний для платформ Microsoft Windows та Mac OS. Перше видання пакету Microsoft Office відбулося у 1990 році. З того часу програмний пакет майже щорічно оновлюється і отримує новий функціонал.

За допомогою програми для роботи з електронними таблицями Microsoft Office Excel ми можемо провести статистичний аналіз даних [8]. Зокрема, побудувати лінійну регресію. Але функція регресійного аналізу була додана до пакету програми в пізніх версіях. Тому для використання її повного функціоналу потрібна інструкція. І в будь-якому разі функціоналу Excel буде недостатньо поставлених вище задач. Загалом все сімейство програм Microsoft Office призначено для розв’язання великого спектру завдань. Але всі вони відносяться до більш простих. Тому Microsoft Office Excel не допоможе вирішити проблему ефективного менеджменту хронічних захворювань.

Ціна останньої версії базового пакету програм складає приблизно 120 доларів США. Це є ще одним фактором, через який сімейство програм Microsoft Office важко розглядати як систему для вирішення задачі описаної вище, адже впровадження цього програмного забезпечення потребує відносно значного фінансування.

2.2 Програмний продукт MatLab

Пакет прикладних програм для чисельного аналізу сімейства MatLab має більше, ніж мільйон користувачів на виробництвах та серед науковців. Він був

створений компанією The MathWorks. Перша версія була випущена у 1984 році, після чого отримала багато оновлень.

Мова програмування, що використовується в даному пакеті, робить його зручним засобом для роботи з математичними матрицями, малювання функцій, роботи з алгоритмами, створення робочих оболонок з програмами в інших мовах програмування. Спеціальні засоби, які працюють з програмним забезпеченням Maple роблять його повноцінною системою для роботи з алгеброю [9].

Але складність у використанні для користувача, що має невеликий досвід роботи з комп'ютерними системами та ціна понад 2000 доларів США за базову комерційну версію без багатьох корисних інструментів роблять його не найкращим варіантом системи, яка буде інтегруватися у заклади, що займаються аналізом медичних даних.

А складність інтеграції його до інших мов програмування може створити ситуацію неправильного вирішення питання ефективного менеджменту хронічних захворювань. Тому програмне забезпечення сімейства програмних продуктів MatLab не дає змоги повного розв'язання завдань, описаних вище.

2.3 Програмний продукт MathCad

Система комп'ютерної алгебри MathCad, орієнтована на підготовку інтерактивних документів з обчисленнями і візуальним супроводженням. MathCad був задуманий і створений Аленом Раздовим з Массачусетського технологічного інституту. Перша версія програми була випущена у 1984 році. Після цього вона постійно оновлюється [10].

Програма MathCad надає можливість працювати зі статистичними даними. В ньому представлений широкий спектр функцій для роботи з різними статистичними вибірками, зокрема, реалізовано регресійний і кореляційний аналіз.

Але функціонал MathCad також не дозволяє створити на його базі систему менеджменту хронічних захворювань. Це програмне рішення підтримує формати MathCad Document, файли формату XML та файли бінарного формату для читання

та запису. Також сам інтерфейс може бути незрозумілим для недосвідченого комп’ютерного користувача. А ціна в 700 доларів США на MathCad в базовій комплектації є надто високою для підприємств, які займаються аналітикою в медичній галузі.

Висновки до розділу

У даному розділі було оглянуто теоретичні та практичні рішення про проблему ефективного менеджменту хронічних захворювань, які використовуються в цій області. Окрім цього, також було досліджено вже існуючі на даний момент програмні рішення, описано можливості їх використання та недоліки цих програмних продуктів.

3 ЗАСОБИ РОЗРОБКИ

Важливим чинником, під час розробки програмного продукту, є вибір засобів програмної реалізації та технологій.

Середовищем розробки інформаційної системи для ефективного менеджменту хронічних інфекційних захворювань було обрано Microsoft Visual Studio Community 2017.

Для створення графічного інтерфейсу інформаційної системи використовувався інтерфейс програмування додатків Windows Forms, що є частиною Microsoft .NET Framework. Він спрощує доступ до елементів графічного інтерфейсу операційної системи Microsoft Windows створюючи обгортку і завертаючись до неї за допомогою керованого коду.

Для розробки алгоритмів використовувалась об’єктно-орієнтована мова програмування C#, з безпечною системою типізації для платформи Microsoft .NET. Цю мову програмування було обрано завдяки її універсальності та зручності у використанні.

Для зчитування вхідної інформації в систему та створення звітів у форматі Excel документа була використана бібліотека Microsoft.Office.Core та Microsoft.Office.Interop.Excel.

Для побудови графіків і діаграм та їх відображення у системі була використана бібліотека System.Drawing.

3.1 Середовище розробки Visual Studio 2017

Середовище розробки Visual Studio 2017 дозволяє ефективно, якісно і швидко писати код, при цьому не втрачаючи уваги з контексту поточного файлу. За допомогою Visual Studio можливо легко переглянути структуру виклику та пов’язані

функції, повернені значенні та стан тестування. Також можна провести рефакторинг коду, знайти помилки та отримати варіанти їх усунення [11].

На сьогоднішній день продукти Microsoft з сімейства Visual Studio містять в собі багатий інструментарій, який знадобиться програмістам для розробки програмних продуктів різного рівня складності. Сімейство Visual Studio 2017 містить в собі такі засоби розробки:

– інтегроване середовище розробки;

– мультиплатформний редактор коду Visual Studio Code, який доступний для систем на базі операційних систем Windows, Mac та Linux;

– зручний сервіс для організації командної роботи під назвою Visual Studio Team Services.

Також ведеться розробка рішення для користувачів під платформою Mac OS – Visual Studio for Mac. Вже сьогодні цей програмний продукт використовує кілька великих світових компаній, які займаються розробкою інформаційних систем та програмних додатків.

З кожним роком інструментарій Visual Studio оновлюється і постійно розширюється. Компанія Microsoft постійно намагається покращити зручність середовища розробки, з урахуванням побажань розробників програмних продуктів з усього світу, які користуються Visual Studio. В результаті середовище розробки Microsoft Visual Studio має загально більше ніж 30 мільйонів завантажень на сьогоднішній день, що робить цей продукт одним із найпопулярніших серед розробників додатків.

Кожен з розробників може завантажити собі 60-денну безкоштовну версію Visual Studio 2017 і повністю ознайомитись з повним функціоналом середовища розробки, а також протестувати його на власному досвіді та враховуючи свої власні вимоги до середовища розробки і зробити для себе певні висновки. Після чого, в залежності від потреб, лишитися на версії Community, якщо інструментарій використовується лише для освітніх цілей, або перейти на платну версію в тому разі, коли розроблений програмістом додаток буде використовуватись як комерційний продукт.

Версія Professional буде корисна для невеликих команд або людей, що займаються розробкою програмних продуктів самостійно і не потребують дуже широкого функціоналу від середовища розробки.

А версія Enterprise ідеальний варіант для великих компаній і команд розробників, де окрім програмування системи потрібні різні функції менеджменту якості та контролю процесів. У Visual Studio Enterprise з'являються такі модулі як:

– імітаційне модулювання;

– модуль "Система менеджменту якості";

– редактор класів і параметрів MetaEdit;

– модуль "Контроль процесів";

– модуль "BUSINESS STUDIO PORTAL";

– формування і оновлення бази знань компанії розробника, що містить всю необхідну регламентуючу документацію та звіти;

– доступ до бази знань з підтримкою розділу прав та зворотного зв'язку;

– контроль показників та досягнення цілей.

Враховуючи потрібні розробнику, команді розробників або цілій компанії функції, вони мають можливість обрати версію, що відповідає ним.

Також варто виділити і модульний підхід до процесу установки інструменту. Серед великої кількості різних модулів, що дозволяють програмістам розробляти додатки різного рівня і для різних платформ, тепер можна обрати лише ті, що потрібні конкретно для реалізації даної задачі. Це прискорює процес установки, а отже і загальний час, що витрачається на розробку програмного продукту загалом, та збільшує загальну ефективність команди або компанії.

Ще одним важливим нововведенням для сімейства Microsoft Visual Studio стала інтеграція з хмарними сервісами платформи Azure. Ця інтеграція дозволяє полегшити створення, налаштування і публікацію розроблених програмних рішень в хмарі Azure напряму із середовища IDE.

У листопаді 2016 року почалася розробка Visual Studio for Mac. На даний момент вже відбулося декілька ранніх випусків цього середовища розробки для Mac OS від Microsoft. Також варто зазначити, що інтегроване середовище розробки

Visual Studio for Mac вже використовується декількома великими компаніями, які займаються розробкою програмних продуктів. Це середовище характеризується розробкою додатків для смартфонів, хмарних рішень, а також програмних рішень для операційної системи Mac OS. Компанією ведеться постійна робота над оптимізацією та виправленням помилок, з урахуванням побажань розробників. Вже були додані підтримка NuGet та .NET Core.

3.2 Інтерфейс Windows Forms

Інтерфейс програмування додатків Windows Forms, який відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. За допомогою цього інтерфейсу, в середині середовища розробки ми можемо отримати доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого додатку в керованому коді. А за рахунок того, що керований код – це класи, що реалізують додаток Windows Forms і вони не залежать від мови розробки, програміст може спокійно використовувати Windows Forms при підготовці програмного забезпечення як на C++, C#, так і на J# або VisualBasic.Net та інших мовах [12].

В наш час Windows Forms використовується як замінник старої і складнішої бібліотеки MFC, що була написана на мові програмування C++. Але розглядаючи Windows Forms варто зазначити, що вона не пропонує парадигму, як це робила MFC.

Виправити цю ситуацію покликані сторонні бібліотеки, які існують у Windows Forms. Головною є бібліотека User Interface Process Application Block, що використовується найчастіше серед розробників додатків на базі Windows Forms. Вона була випущена групою програмістів Microsoft, які займаються саме прикладами та рекомендаціями. User Interface Process Application Block безкоштовна та доступна для скачування з офіційного сайту. Разом з нею можна знайти вихідний код та приклади, які покращать розуміння бібліотеки та прискорять навчання.

Додаток Windows Forms – дієво-орієнтовний, а отже виконання програми будується на певних подіях, які підтримуються Microsoft .NET Framework. Це можуть бути дії користувача, повідомлення від інших додатків і програм, події операційної системи та інше. Це спрощує розуміння логіки вже готового програмного продукту, а отже і процес роботи з програмним продуктом взагалі, що є плюсом для користувачів.

Цей інтерфейс містить в собі велику кількість платних та безкоштовних компонентів для розробки програм, побудованих на Windows Forms [13].

До безкоштовних компонентів Windows Forms відносяться такі групи контролерів:

– Chart Controls;

– SharpPieces;

– Ajax Control Toolkit.

Цих контролерів достатньо для розробки програмних продуктів та інформаційних систем з відносно простим інтерфейсом користувача. Для реалізації програмного продукту за завданням, описаним вище, запропонованих компонентів буде достатньо.

Для розробки більш складних інформаційних систем, програміст або команда програмістів може звернутись до платних компонентів розробки графічного інтерфейсу. До них можна віднести компоненти від таких компаній розробників програмних продуктів як:

– DevExpress;

– ComponentOne;

– Dundas;

– Telerik;

– Obout.

Набор компонентів для розробки графічних інтерфейсів, запропонований цими компаніями дуже великий. Його буде достатньо для розробки інтерфейсів високої складності[14].

3.3 Бібліотека Microsoft.Office.Interop.Excel

Інструментарій Microsoft Visual Studio і мови програмування C# передбачає функції, що спрощують доступ до об’єктів сімейства Microsoft Office [15]. Групою таких об’єктів надає Microsoft Office Excel. Він має велику кількість об’єктів, з якими можна взаємодіяти.

Інтерфейс Application – основний інтерфейс в компонентному класі COM, що необхідний для цього керованого коду для взаємодії з відповідним COM-об’єктом. Цей інтерфейс використовується тільки тоді, коли метод, який буде використано сумісно, буде схожий з іменем події. В цьому випадку цей інтерфейс перетворюється для виклику метода. В іншому випадку використовується інтерфейс Microsoft .NET, що є похідним від компонентного класу COM [16].

Бібліотека Microsoft.Office.Interop.Excel.Workbook – це центральний об’єкт для програмування в Excel. Він представляє собою сторінку документа Excel, а також весь її зміст. Якщо ви спробуєте відкрити вже існуючий документ або створити новий, то таким чином ви створюєте новий об’єкт Microsoft.Office.Interop.Excel.Workbook. Він в свою чергу буде доданий до колекції T:Microsoft.Office.Interop.Excel.Workbook об’єкта Application. Документ, який знаходиться у роботі в даний момент часу називають активним документом, а лист документа Excel – активним листом. Він представлений властивістю P:Microsoft.Office.Interop.Excel._Application.ActiveSheet об’єкта Application.

Бібліотека Microsoft.Office.Interop.Excel.XlCellType – це об’єкт, що також використовується для програмування в Excel. Він відповідає за вказання типу комірок в документі Microsoft Office Excel. В свою чергу він має потрібні нам параметри комірок. В цьому випадку нас цікавлять дві його властивості, а саме – lastCell.Row та lastCell.Column, які передаються до колекції T:Microsoft.Office.Interop.Excel.XlCellType об’єкта Application. За допомогою цих властивостей ми можемо здійснювати навігацію по сторінці активного документа Excel. Розробнику додатку це потрібно для того, щоб контролювати з якої комірки

він отримує данні та в яку, в свою чергу, записує їх (наприклад, при формуванні звіту).

3.4 Бібліотека System.Drawing

Графічний інтерфейс програмування Windows Forms, що є додатковим інструментом середовища розробки Microsoft Visual Studio потребує певного підключення до мови C#. Щоб зробити це підключення та зробити цей елемент керованим кодом потрібно використати бібліотеку System.Windows [17].

Простір імен System.Drawing забезпечує доступ до функціональних можливостей графічного інтерфейса програмування Windows Forms. Він містить в собі такі простори імен як:

– System.Drawing.Drawing2D;

– System.Drawing.Imaging;

– System.Drawing.Text.

Ці простори імен відповідають за додаткові функціональні можливості цієї бібліотеки.

Загалом, бібліотека System.Drawing має потужний набір класів, що повністю відповідають за графічну реалізацію продукту, що розроблюється на основі мови програмування C#.

Наприклад, клас Graphics в свою чергу дає доступ розробнику програмного продукту до методів, які відповідають за малювання та відображення на дисплеї пристрою. Він інкапсулює поверхню малювання графічного відображення інтерфейсу. Цей клас ніяк не унаслідується.

Класи Rectangle та Point інкапсулюють елементи графічного інтерфейсу розробки.

Клас Pen використовується для малювання різного роду ліній та кривих, що ми можемо побачити у Windows.Forms. Цей клас визначає об’єкт, який відповідає за ці функції. Він, як і клас Graphics, також не успадковується.

Клас Brush відповідає за визначення об’єктів, які використовуються для заповнення кольором всередині різних графічних фігур. Це можуть бути такі елементи графічного інтерфейсу Windows Forms:

– прямокутник;

– еліпс;

– круг;

– багатокутник;

– різного роду доріжки [18].

Але клас Brush, на відміну від класів Graphics, Pen, Rectangle та Point, має певну особливість. Це абстрактний базовий клас, який не може бути реалізованим.

Для створення об’єкту класу Brush використовуються класи, що є похідними від класу Brush, а саме:

– TextureBrush;

– SolidBrush;

– LinearGradientBrush.

Клас Bitmap інкапсулює точковий малюнок графічного інтерфейсу користувача, що складається з пікселів графічного зображення та атрибутів малюнка. Тобто об’єкт класу Bitmap використовується для роботи з зображеннями, що описуються виключно даними пікселей. Без використання цього класу розробник програмного продукту або інформаційної системи просто не зміг би зберегти певні графіки або створити скріншот роботи програми з середини для формування певного користувацького звіту [19].

Висновки до розділу

У даному розділі було оглянуто основні засоби розробки програмного продукту за заданою темою. Були описано середовище розробки системи, основні бібліотеки, що були використані для забезпечення роботи програмного додатку, а також графічний інтерфейс програмування.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі міститься перелік використаних обчислювальних методів. А також опис використаних обчислювальних методів і математичні та програмні засоби їх реалізації.

4.1 Методи імітаційного моделювання

Використання індивідуально-орієнтовних моделей в обґрунтуванні фармацевтичного забезпечення діагностичного скринінгу та вакцинопрофілактики хронічних вірусних захворювань в популяції, в якій окремі особи можуть бути віднесені до різних епідеміологічних класів. При цьому потрібно враховувати, що для кожної хворої особи враховується прогресування вірусного захворювання з часом. Така модель є об'єднанням епідеміологічних моделей та може бути використана для оптимізації діагностик і вакцинопрофілактики хронічних захворювань за такими критеріями:

– загальному зменшенні кількості осіб в певному епідеміологічному чи клінічному класі по відношенню до можливих втручань протягом визначеного періоду часу;

– отриманні максимального фармакотерапевтичного ефекту [20].

Індивідуально-орієнтовна модель – це обчислювальний інструмент, що дозволяє проводити моделювання беручи до уваги індивідуальні особливості та взаємодію агентів в системі, що розглядається ІОМ дозволяє дослідити, як властивості системи виникають на основі адаптивної поведінки агентів, а також як, з іншого боку, система впливає на усіх агентів.

1. Нехай кожний окремий агент описується набором своїх характеристик

$$I_{n,t} = [C_{n,1,t}, C_{n,2,t}, \dots, C_{n,m,t}], \quad (1)$$

де n – послідовний номер, який ідентифікує агенту, m – кількість характеристик агенту, t – момент часу коли агент описується даним набором характеристик.

Для представлення епідеміологічної системи, принаймні одна з характеристик повинна відображати епідеміологічні класи, як це представлено в класичних епідеміологічних моделях, тобто класи сприятливих, інфікованих агентів тощо. В залежності від класу, в якому знаходиться агент, в якості першої характеристики йому присвоюється відповідне натуральне число.

Аналогічним чином визначаються інші характеристики, що визначають статус агенту по відношенню до можливих втручань, наприклад обстежений, діагностований, такий що отримує фармацевтичну допомогу тощо.

2. На кожному інтервалі часу Δt характеристики агенту можуть змінюватися. Вони пов’язані з епідеміологічним класом та статусом агенту на кожному кроці розглядаються як марківський ланцюг. В кожен момент часу агент має певні ймовірності переходу в інший епідеміологічний клас, чи зміни статусу, відмінного від попереднього, а також ймовірність залишитись в нинішньому статусі, що можуть змінюватись в процесі еволюції системи.

Вік агенту на кожному кроці збільшується на Δt.

Всі інші характеристики, крім віку можуть змінюватись, за певними законами розподілу ймовірностей.

3. При введені агенту в систему визначається середня тривалість його життя за експоненціальним законом розподілу ймовірностей. Коли вік агенту досягає тривалості його життя, агент виводиться з системи.

4. На популяційному рівні зберігаються значення розподілу популяції за всіма епідеміологічними класами та статусом, тобто кількість агентів в системі в кожному стані.

Для нашої системи індивідуально-орієнтовна модель має такий вигляд (рисунок 4.1):

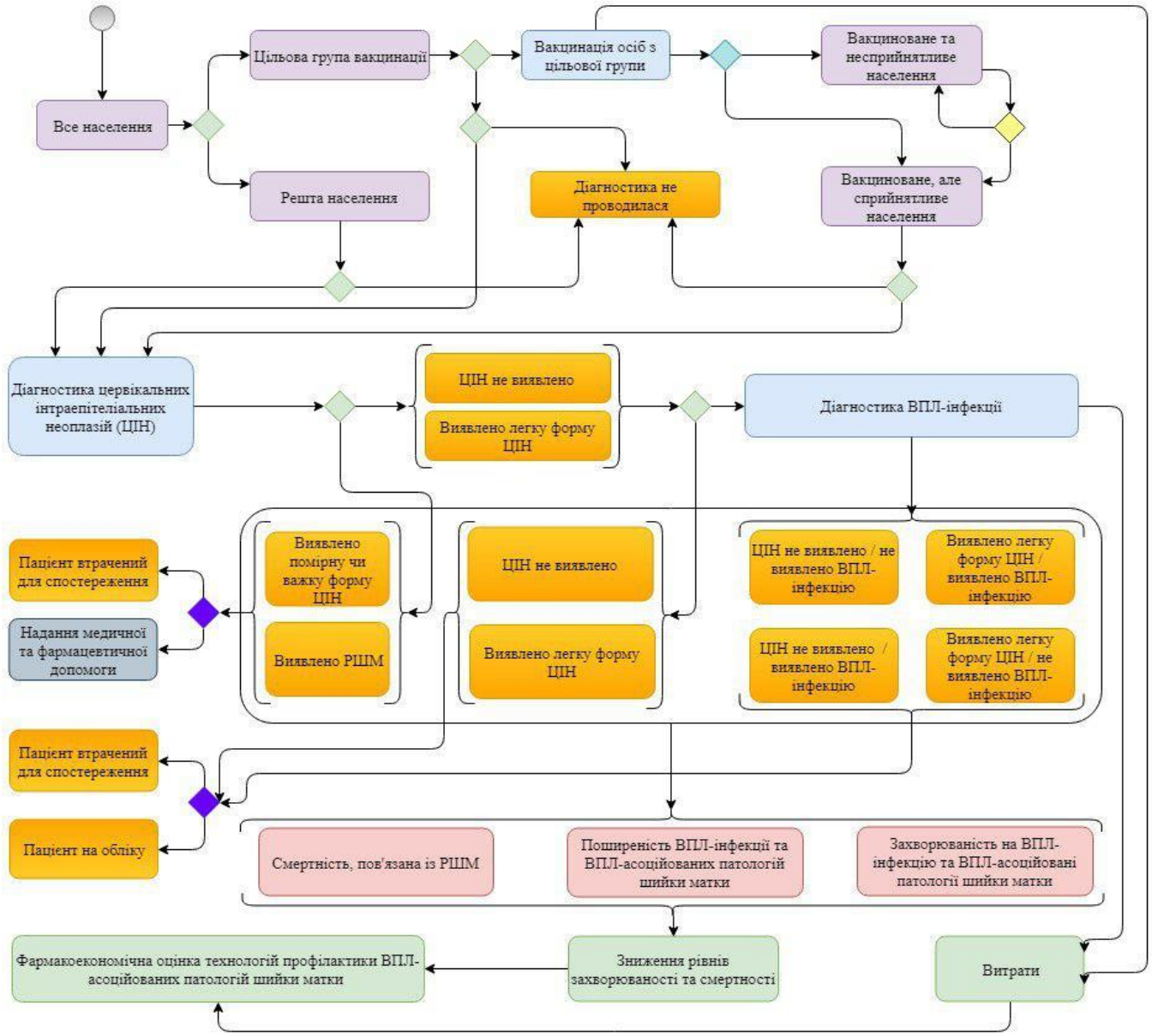


Рисунок 4.1 – Індивідуально-орієнтовна модель

Вище наведена UML діаграма континууму догляду за вірусом папіломи людини у Ванкувері.

Прямокутники з закругленими кутами – події. Зелені ромби представляють собою точки прийняття рішень, а фіолетові ромби – злиття.

Червоні прямокутники являють собою результати епідемії континууму догляду. Особи можуть загинути і залишити систему на будь-якому етапі. Фіолетові прямокутники представляють популяцію на певних її етапах. Блакитні прямокутники являють собою певні фармакотерапевтичні дії, що застосовуються відносно досліджуваної групи. Жовті прямокутники демонструють певний стан, в якому знаходиться особа, після фармакотерапевтичного втручання або прийняття рішення. Зелені прямокутники відповідають за показники витрат, ефективності та оцінки технологій профілактики вірусу папіломи людини.

4.2 Метод Монте-Карло

Методи Монте-Карло – це широкий клас обчислювальних алгоритмів, які засновані на повторювані генерації випадкової вибірки для отримання чисельних результатів. Їх основною ідеєю є використання випадковості для вирішення проблем, які можуть бути детерміністичними в принципі. Вони часто використовуються в фізико-математичних задачах і є найбільш корисними, коли важко або неможливо використовувати інші підходи [21]. Методи Монте-Карло переважно використовуються в трьох проблемних класах: оптимізація, чисельне інтегрування та генерація рисунків з розподілу ймовірності.

В принципі, методи Монте-Карло можуть бути використані для вирішення будь-якої проблеми, що має імовірнісну інтерпретацію. За законом великих чисел, інтеграли, що описуються очікуваним значенням деякої випадкової величини, можуть бути апроксимовані, беручи емпіричне значення незалежних вибірок змінної. Коли розподіл імовірності змінної параметризується, математики часто використовують методи Монте-Карло. Головною ідеєю є розробка розумної моделі марківського ланцюга з заданим стаціонарним розподілом імовірності. Тобто вибірки, що генеруються даним методом, будуть згенеровані з бажаного (цільового) розподілу [22].

4.3 Марковський процес прийняття рішень

Марковський процес прийняття рішення (МППР) – це метод для опису та аналізу послідовних рішень в умовах невизначеності (рисунок 4.2). Він структурно схожий на марківський ланцюг за винятком того, що матриця переходів залежить від дій або стратегій особи, що приймає рішення, в кожний момент часу. Моделі МППР також відомі як послідовні стохастичні оптимізації, дискретно-часові процеси стохастичного керування, стохастичного динамічного програмування тощо. Метою особи, що приймає рішення, є визначення стратегій, при якій серія рішень є оптимальною.

Цей підхід знайшов місце в дослідженні операцій, починаючи з 60-их років 20-го сторіччя [23]. МППР був застосований в широкому діапазоні областей, таких як управління запасами, фінансами, мереж зв'язку, водними ресурсами, теорії ігор, машинного навчання і нещодавно в охороні здоров'я.

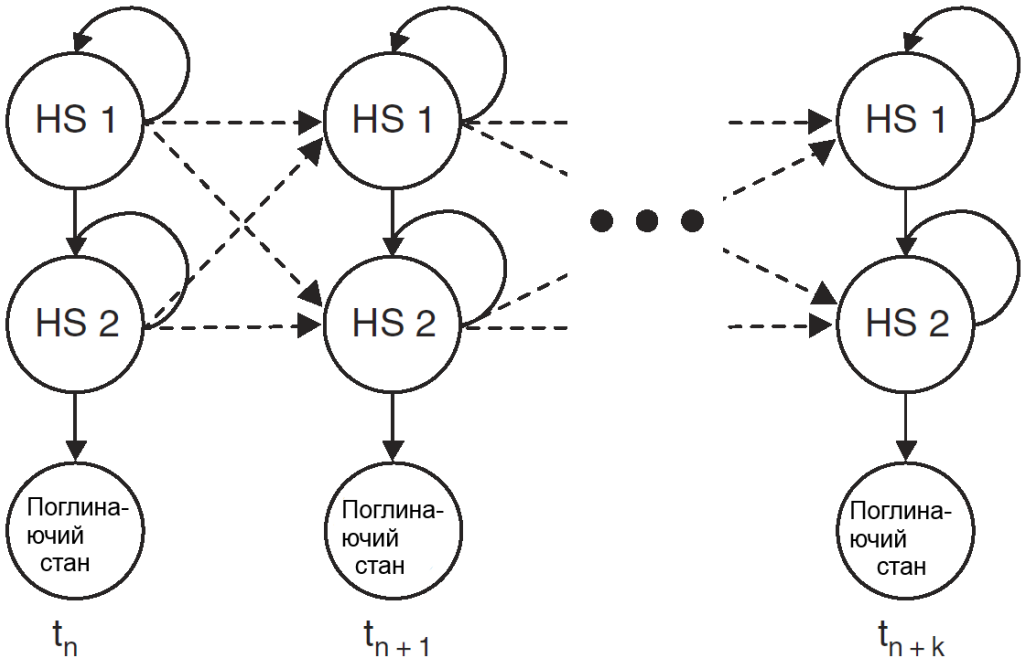


Рисунок 4.2 – Марковський процес прийняття рішень

У марковському процесі прийняття рішень, дії особи, що приймає рішення, задаються стратегією, яка визначає реакцію на будь-які можливі стани системи в

кінці кожного часового інтервалу за весь часовий горизонт. Вибір дій залежить від виграшу в кінці кожного часового інтервалу. Тобто чистого прибутку попередньої ітерації.

Кожна дія має структуру винагороди, таку як очікуване значення результату рішення в стохастичній марковській моделі. У часі t_{n+1} стан системи змінюється. Ця зміна є імовірнісною і марковською, тобто, новий стан системи заснований виключно на стані системи в часі t_n. Метою марковського процесу прийняття рішень є визначення оптимальної стратегії, яка максимізує винагороду або оптимум серед одразу декількох параметрів, таких як тривалість життя та витрати, які впливають на ефективність системи.

Розробник марковського процесу прийняття рішень може скористатися тим, що правила прийняття рішення можуть бути динамічно запрограмованими. Крім того, відповідні правила можуть бути змінені на основі зворотних зв’язків, як і на основі отриманих виграшів в кінці кожного циклу для пошуку оптимальних рішень під час процесу.

Особливим випадком марковського процесу прийняття рішень є стаціонарність стратегії прийняття рішення, яка являється стаціонарною. Це відповідно означає, що правила прийняття рішень залишаються постійними протягом усього часу та не змінюються ні яким чином. марковський процес прийняття рішень з такою стратегією називається марківським ланцюгом.

Одним з ключових припущень, зроблених в марковському процесі прийняття рішень є те, що особа, яка відповідає за прийняття рішення, повинна обрати стратегію в рамках всього часового горизонту моделювання. Більшість справжніх осіб, що приймають рішення не схильні до такої поведінки.

Динамічне програмування може допомогти визначити оптимальну стратегію прийняття рішення і більш адекватну модель поведінки реальних осіб, які приймають рішення (тобто адаптації існуючої стратегії, щойно нова інформація стає доступною).

Для випадку інформаційної технології для ефективного менеджменту хронічних захворювань марківський ланцюг прийняття рішень матиме дещо

спрощений вигляд (рисунок 4.3). В нього будуть входити п’ять вузлів, що характеризують різні стани захворювання:

– здоровий;

– легка форма дисплазії (Cin I);

– помірна та важка форми дисплазії (Cin II-III);

– РШМ;

– смерть.

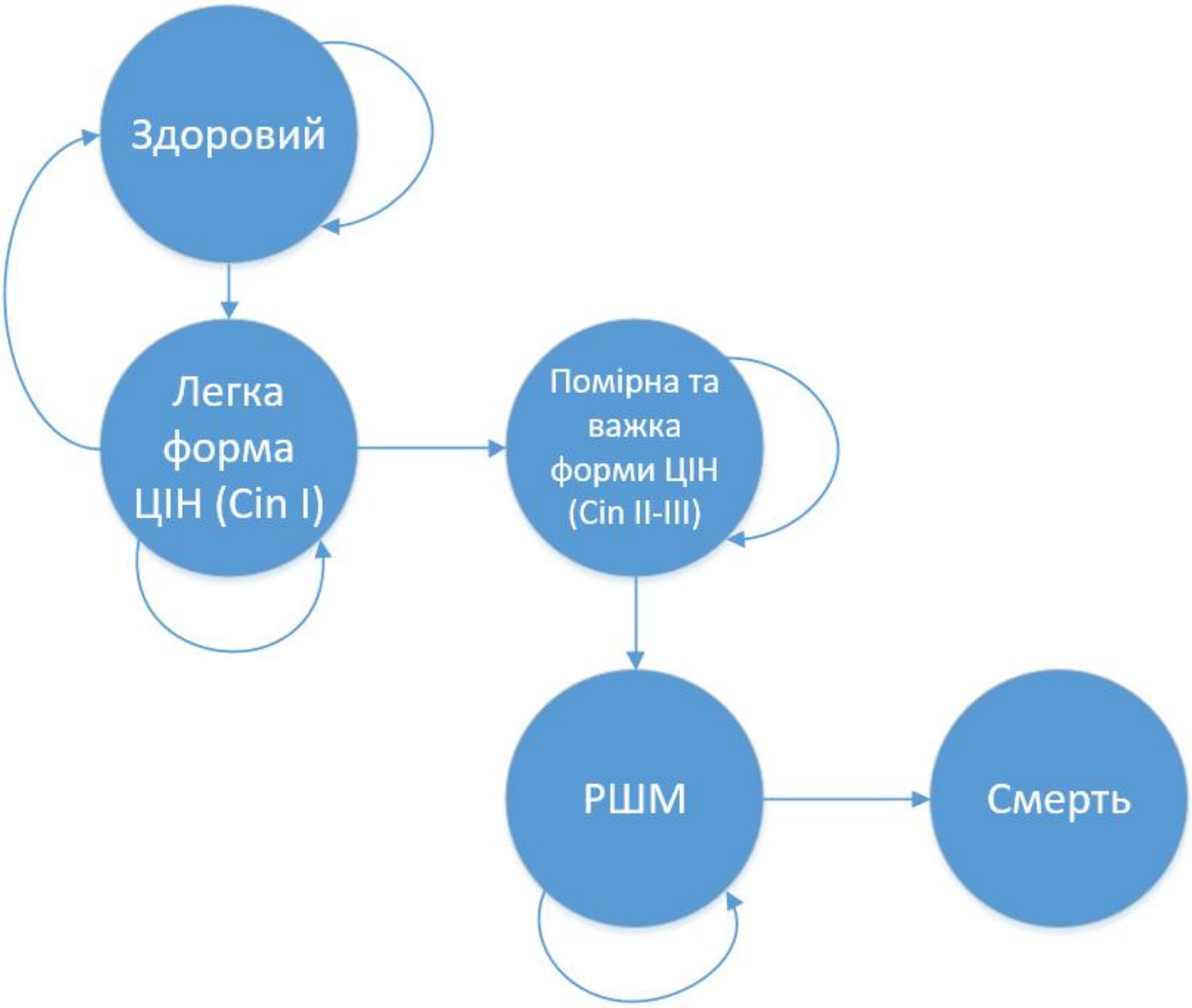


Рисунок 4.2 – Марківський ланцюг для даної системи

Тим не менш, ці ж адаптивні методи, за допомогою наприклад, нейронних мереж, можуть приховувати оптимальний результат для даного набору правил, які

можуть зруйнувати основну мету використання моделі. І, нарешті, в даний час обмеженою є кількість доступного програмного забезпечення, готового до використання менш досвідченими користувачами.

Висновки до розділу

У даному розділі було оглянуто програмні засоби розв’язання задачі ефективного менеджменту хронічних захворювань. А також розроблених алгоритмів, відповідно до поставленої задачі.

5 ОПИС ВИПРОБУВАННЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

Програмний продукт було розроблено з використанням технологій, які підтримуються операційними системами Microsoft Windows і тому він працює на усіх відповідних системах, а саме Windows 10, Windows 8, Windows Vista та Windows XP.

Для забезпечення повноцінної роботи та досягнення високої точності роботи створених додатків для ефективного менеджменту хронічних захворювань було обрано середовище розробки Microsoft Visual Studio 2017, яке показало себе як надійне, зручне та гнучке для розробника рішення.

Також використовувалися додаткові бібліотеки Microsoft.Office.Interop.Excel для роботи з документами програми Microsoft Office Excel та System.Drawing – для роботи з графічним інтерфейсом.

В цьому розділі приведені системні вимоги для роботи з додатком та сценарії роботи користувача з ним.

5.1 Системні вимоги

Для забезпечення коректної роботи, без помилок та сповільнень, інформаційної системи для ефективного моніторингу хронічних інфекційних захворювань персональний комп’ютер повинен мати процесор не гірше, ніж Intel ® Pentium ® / Celeron ® / Xeon™ або краще, та з тактовою частотою не менше 1,8 GHz або більше. Та не гірше, ніж AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron або краще, для користувачів систем на базі процесорів від фірми AMD.

Також комп’ютеру користувача повинно бути доступно не менше 2 Gb оперативної пам’яті.

Оскільки система не виконує ресурсомістких графічних перетворень або роботи з різними моделями, до графічного ядра також немає. Достатньо, щоб воно було не гріше, ніж інтегроване Intel ® HD Graphics 2000, що еквівалентно дискретним графічним картам з об’ємом пам’яті не менше, ніж 128 Mb.

Також на жорсткому диску повинно бути не менше, ніж 30 Mb вільного місця для завантаження розробленого програмного додатку. Цей об’єм вільного місця на жорсткому диску буде також використано для збереження окремих графіків у форматі .png та документу Excel зі звітом, який генерується додатком під час роботи.

На комп’ютері повинна бути встановлена операційна система Windows 10, Windows 7, Windows XP Service Pack 2 або Windows Vista Service Pack 2 від компанії Microsoft.

Користувачу також знадобиться програма із сімейства Microsoft Office, а саме Microsoft Office Excel або будь-яка інша, здатна генерувати документи у форматах .xls або більш новому – .xlsx.

Вона потрібна для створення файлу з вхідними даними, які буде обробляти інформаційна система. Звіт з роботи програми також генерується у форматі Excel документу, тому Microsoft Office Excel або інша програма, здатна відкривати файли, вказаного вище формату, знадобиться і для ознайомлення з ним.

Для перегляду збережених графіків за межами програми ніякі додаткові програми не потрібні. Стандартного переглядача зображень, що доступний кожному користувачу систем на основі операційної системи Windows за замовчуванням – достатньо.

5.2 Робота користувача з програмним продуктом

Програмний продукт не потребує інсталяції додаткових програм. Тому для використання, користувачу потрібно тільки запустити файл з назвою “WindowsFormsApp1.exe”.

Після чого запуску програмного додатку, користувач побачить інтерфейс Windows Forms (рисунок 5.1).

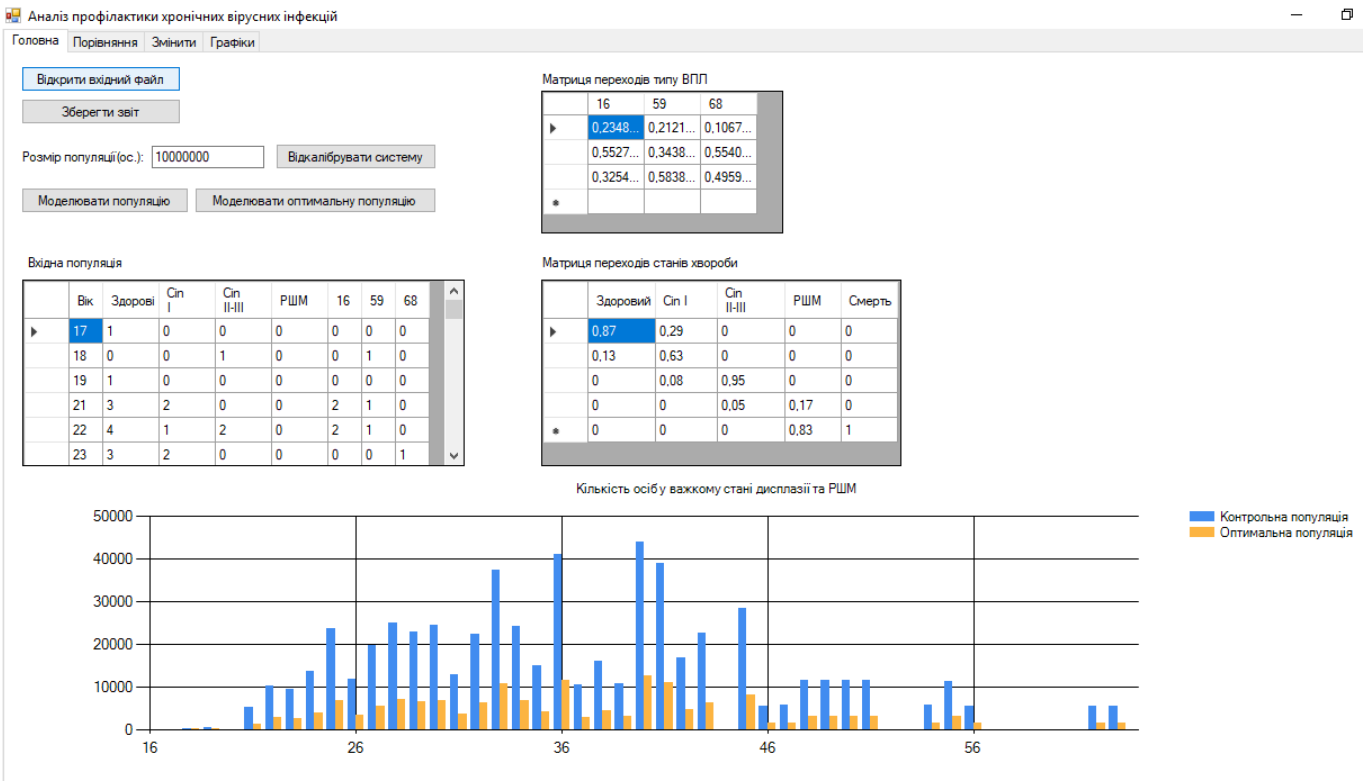


Рисунок 5.1 – Головне вікно додатку Windows Forms

Весь інтерфейс розділено на чотири головні вкладки, а саме:

– “Головна”;

– “Порівняння”;

– “Змінити”;

– “Графіки”.

З їх допомогою відбувається навігація програмним продуктом (рисунок 5.2).

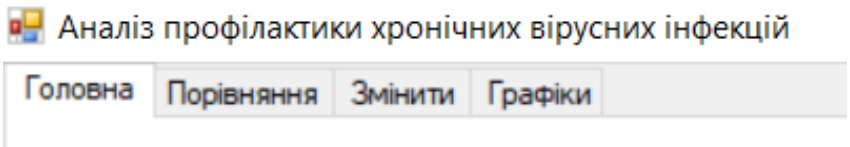


Рисунок 5.2 – Навігаційні вкладки програмного продукту

Перед початком роботи аналітик повинен підготувати Excel документ, який буде містить в собі дані про популяційну групу, на основі якої буде моделюватися вся популяція людей. Ці дані потрібні для коректної роботи інформаційної системи для ефективного менеджменту хронічних інфекційних захворювань. Відповідно до цих даних система буде відкалібрована.

Документ Excel, який містить вхідну інформацію, повинен складатися з восьми колонок, що розставлені в певній послідовності та містять наступні відомості про групу:

– вік;

– кількість здорових людей у популяції;

– кількість людей з дисплазією початкового рівня або CIN I;

– кількість людей з дисплазією середнього рівня або CIN II-III;

– кількість людей з дисплазією важкого рівня або РШМ;

– кількість людей з шістнадцятим типом вірусу папіломи людини;

– кількість людей з п’ятдесят дев’ятим типом вірусу папіломи людини;

– кількість людей з шістдесят восьмим типом вірусу папіломи людини.

Послідовність колонок з інформацією відповідає приведеному вище переліку і повинна зберігатись для коректної роботи програмного додатку. Використовуючи неправильну інформацію, користувач отримає результати, що не відповідають дійсності та не можуть бути використані в подальшому.

Всю цю інформацію користувач інформаційної системи може отримати за відповідним запитом від медичних центрів та медичних закладів, які займаються дослідженнями хронічних захворювань [24].

Перед початком роботи спеціаліст повинен перевірити коректність структури отриманих вхідних даних. Це можна зробити у програмі Microsoft Office Excel або будь-якій іншій програмі, що здатна зчитувати, редагувати та зберігати документи у форматі .xsl або .xslx. Після чого отриманий файл користувач може використовувати у програмному додатку, як вхідний файл.

Для початку роботи користувач повинен натиснути кнопку “Відкрити вхідний файл” (рисунок 5.2). Вона знаходиться на вкладці “Головна” у розробленому програмному продукті.

Рисунок 5.2 – Кнопка для відкриття файлу з вхідною інформацією

Після чого у діалоговому вікні він повинен обрати вхідний файл, який підготував раніше. Програмним продуктом передбачена робота з вхідними файлами, що зберігаються у форматі .xls або .xlsx для новіших версій програми Excel (рисунок 5.3).

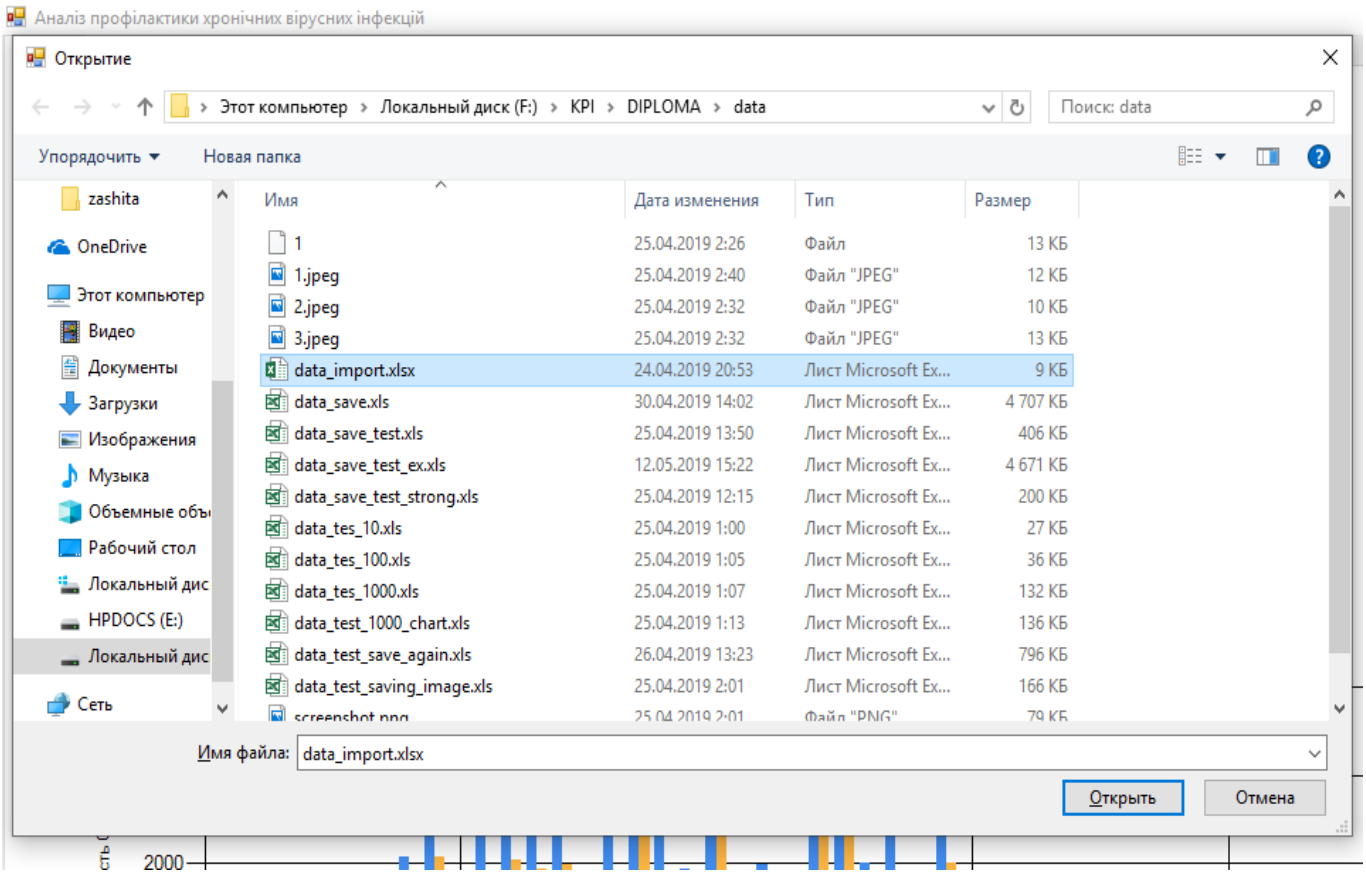


Рисунок 5.3 – Діалогове вікно для вибору файлу з вхідними даними

Файл буде загружено у систему та зчитано дані, що містяться у ньому. Після чого у таблицю з назвою “Вхідна популяція” буде занесена інформація, що містилась у даному файлі.

Для того, щоб користувач міг ще раз затвердити коректність даних, а також у будь-який час ознайомитись з ними (рисунок 5.4). Корегувати вхідну популяцію в середині системи неможливо, щоб не допустити випадкових помилок, які можуть виникнути через неуважність користувача.

Вхідна популяція

	Вік	Здорові	Cin I	Cin II-III	РШМ	16	59	68
▶	17	1	0	0	0	0	0	0
	18	0	0	1	0	0	1	0
	19	1	0	0	0	0	0	0
	21	3	2	0	0	2	1	0
	22	4	1	2	0	2	1	0
	23	3	2	0	0	0	0	1

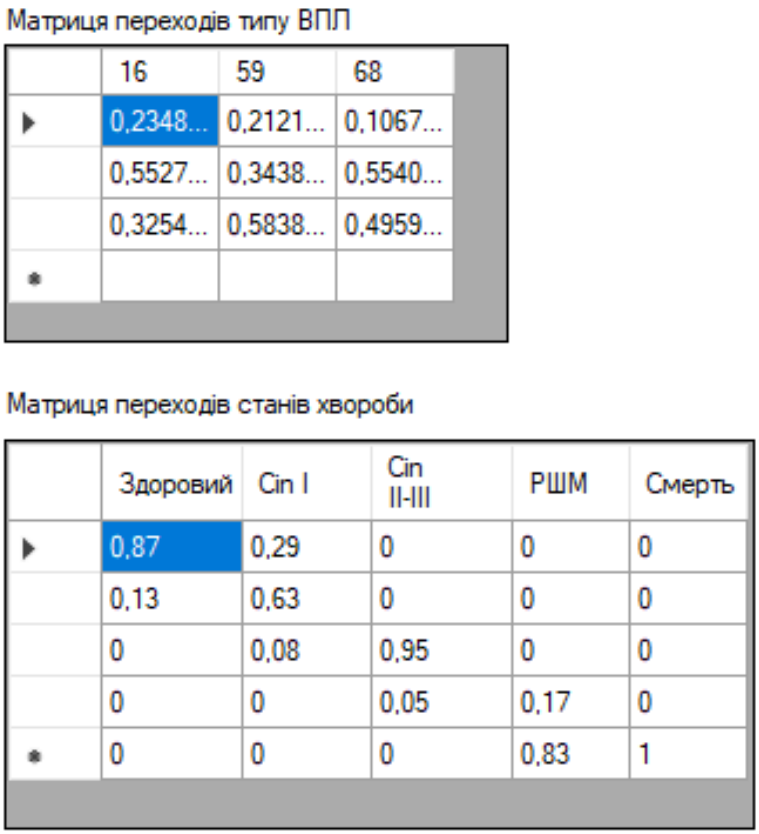
Рисунок 5.4 – Таблиця з даними про вхідну популяцію

Після загрузки вхідної популяції аналітик отримає змогу відкалібрувати систему натиснувши відповідну кнопку на вкладці “Головна” (рисунок 5.5).

Рисунок 5.5 – Кнопка, що відповідає за калібрування системи

За алгоритмом, описаним вище система буде відкалібрована, а користувачу буде представлено матрицю переходів із одного стану хвороби в інший та матрицю переходів за різними станами вірусу папіломи людини. Цю інформацію він має можливість переглянути у таблицях “Матриця переходів станів хвороби” та

“Матриця переходів типу ВПЛ” відповідно (рисунок 5.6). Їх також можна знайти на вкладці “Головна”.



Матриця переходів типу ВПЛ

	16	59	68
▶	0,2348...	0,2121...	0,1067...
	0,5527...	0,3438...	0,5540...
	0,3254...	0,5838...	0,4959...
*			

Матриця переходів станів хвороби

	Здоровий	Cin I	Cin II-III	РШМ	Смерть
▶	0,87	0,29	0	0	0
	0,13	0,63	0	0	0
	0	0,08	0,95	0	0
	0	0	0,05	0,17	0
*	0	0	0	0,83	1

Рисунок 5.6 – Матриці переходів

В цих таблицях виводяться імовірності переходів із одного стану хвороби в інші для кожного з п’яти станів (Здоровий, Cin I, Cin II-III, РШМ та смерть) та імовірності переходів агенту з однієї групи вірусу папіломи людини в іншу (16, 59, 68). Імовірність переходу не може бути менша за нуль – мінімальна та більша за одиницю – максимальна.

Для деяких з переходів із одного стану хвороби в інший, вірогідність переходу за замовчуванням буде рівне нулю та ніяк не змінюється. Це передбачається, тому що деякі переходи зі стану в інший стан просто неможливі у природі за розвитком хронічного інфекційного захворювання у людини, які розглядаються в цьому програмному продукті, а саме:

– перехід зі стану “Здоровий” до стану “Cin II-III”;

– перехід зі стану “Здоровий” до стану “РШМ”;

– перехід зі стану “Здоровий” до стану “Смерть”;

– перехід зі стану “Cin I” до стану “РШМ”;

– перехід зі стану “Cin I” до стану “Смерть”;

– перехід зі стану “Cin II-III” до стану “Здоровий”;

– перехід зі стану “Cin II-III” до стану “Cin I”;

– перехід зі стану “Cin II-III” до стану “Смерть”;

– перехід зі стану “РШМ” до стану “Здоровий”;

– перехід зі стану “РШМ” до стану “Cin I”;

– перехід зі стану “РШМ” до стану “Cin II-III”;

– перехід зі стану “Смерть” до стану “Здоровий”;

– перехід зі стану “Смерть” до стану “Cin I”;

– перехід зі стану “Смерть” до стану “Cin II-III”;

– перехід зі стану “Смерть” до стану “РШМ”.

Після калібрування інформаційної системи у аналітика з’являється можливість моделювати контрольну популяцію. Це можна зробити натиснувши на кнопку “Моделювати популяцію” на вкладці “Головна” (рисунок 5.7).

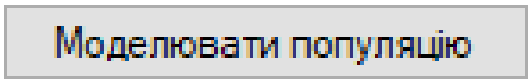


Рисунок 5.7 – Кнопка, яка запускає моделювання контрольної популяції

Також користувач може змоделювати оптимальну популяцію. Це можна зробити натиснувши на кнопку “Моделювати оптимальну популяцію”, що знаходиться поряд (рисунок 5.8).

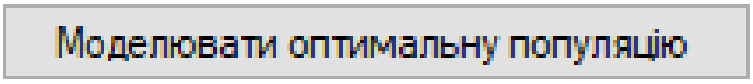


Рисунок 5.8 – Кнопка, яка запускає моделювання оптимальної популяції

Моделюючи контрольну та оптимальну популяції, користувач інформаційної системи має можливість вручну вказати розмір популяції людей, яку він хоче моделювати.

Це робиться у відповідному полі програмного додатку “Розмір популяції (ос.)”, яке знаходиться на вкладці “Головна” (рисунок 5.9). Розмір повинен бути вказаний у кількості осіб. Це потрібно для наближення популяції, що моделюється, до існуючої популяції людей. Наприклад, коли мова йде про населення певного району, міста, області, регіону чи країни і потрібно отримати точні дані про розвиток хронічного захворювання, враховуючи усіх людей, що мають ризик бути інфікованими та тих, кого вже інфіковано.

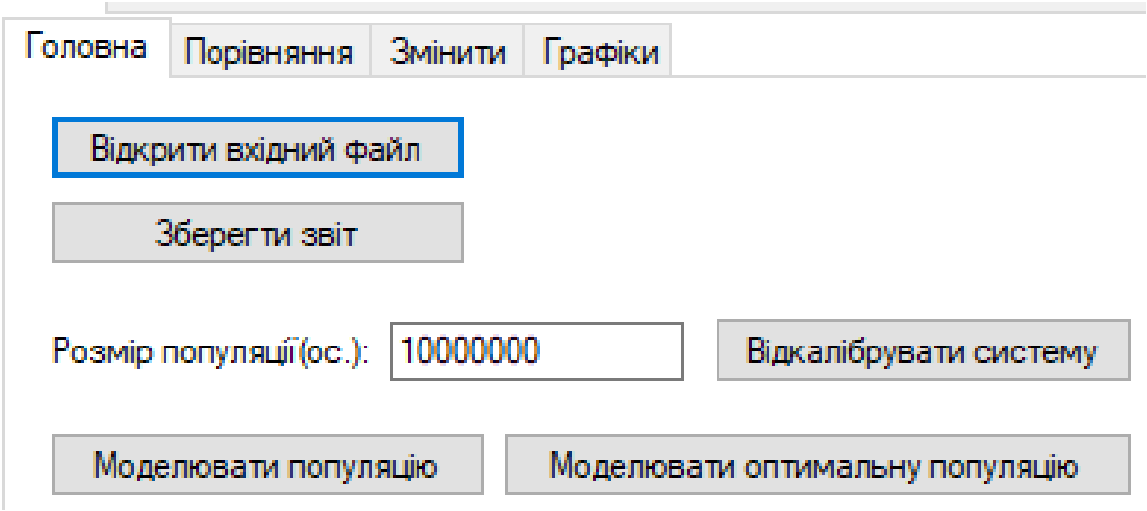


Рисунок 5.9 – Встановлення розміру популяції та її моделювання

Моделюючи обидві популяції, користувач інформаційної системи отримує дані, як буде розвиватися контрольна та оптимальна популяція в наступні три роки.

Аналізуючи ці дані, він може дізнатися яка кількість людей буде лишатися здоровою, яка перейде до станів дисплазії початкового, середнього та важкого рівнів, у якої частини буде виявлено рак шийки матки, а яка частина популяції помре внаслідок прогресування хронічного інфекційного захворювання. Для порівняння ефективності контрольної фармакотерапевтичної стратегії з оптимальною, яку пропонує система, загальна кількість осіб, які знаходяться у

важкому стані дисплазії або мають рак шийки матки, виводиться у графіку, розділеному за віком агентів (рисунок 5.10).

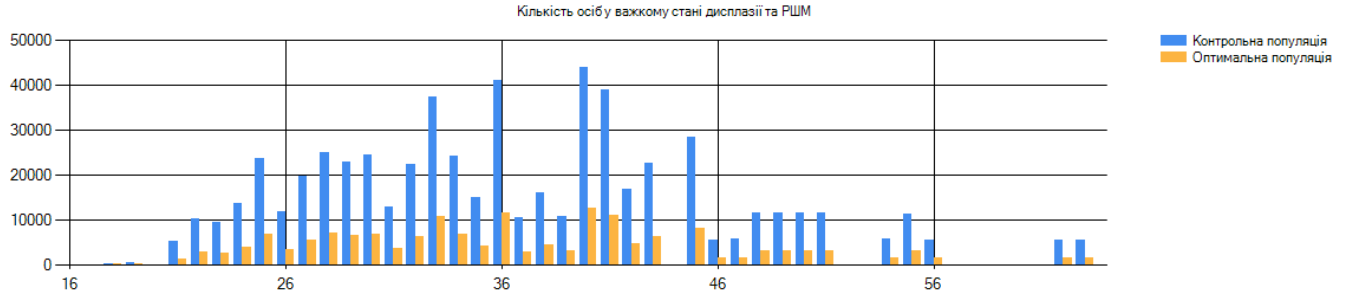


Рисунок 5.10 – Графік порівняння контрольної та оптимальної популяції

Ці данні також виводяться у таблицях “Контрольна популяція” та “Оптимальна популяція” на вкладці “Порівняння” (рисунок 5.11).

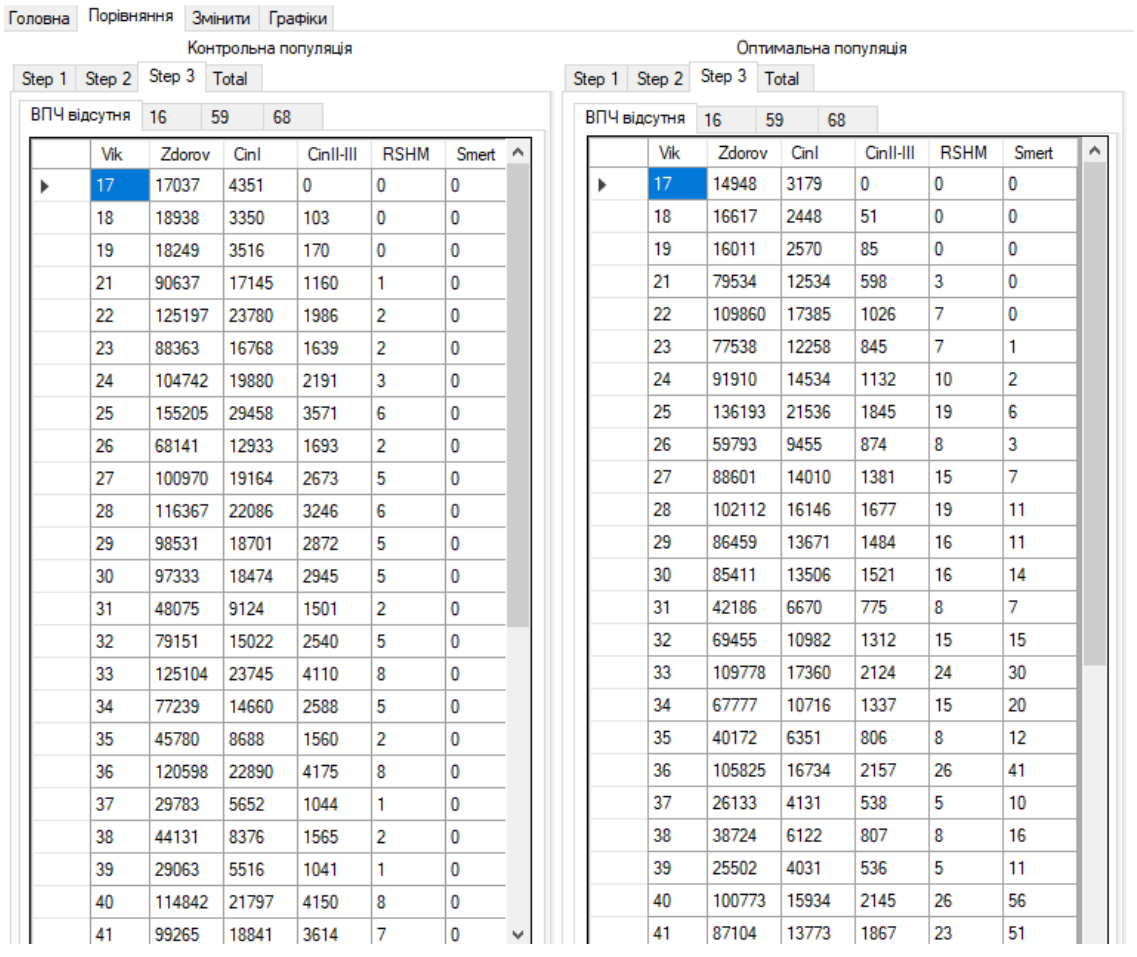


Рисунок 5.11 – Таблиці розвитку популяцій

Ці данні структуруються за типом вірусу папіломи людини (відсутній, шістнадцятий тип, п’ятдесят дев’ятий тип та шістдесят восьмий тип) на кожному з кроків, які виступають у якості одного року моделі, а також за кожною групою по віку, в залежності від того, які були представлені на вході в систему.

На вкладці “Порівняння” також можна побачити дві кругові діаграми. Вони демонструють співвідношення кількості здорових людей популяції (синій сектор діаграми) до кількості людей з важкою стадією дисплазії та померлих (жовтий сектор діаграми) (рисунок 5.12).

Рисунок 5.12 – Діаграми співвідношень здорових людей до людей у важкому стані та померших

Згідно до цієї інформації, користувач може зробити висновки на скільки ефективна запропонована системою стратегія медичного та фармакотерапевтичного втручання у хід розвитку хронічного інфекційного захворювання серед населення [25].

Запропоновані значення вакцинації, першого та другого скринінгів, а також розподілу бюджету між цими заходами можна подивитись у відповідних комірках програмного додатку, що знаходяться на вкладці “Порівняння” (рисунок 5.13).

Рисунок 5.13 – Оптимальні вірогідності проходження певних процедур та оптимальний розподіл бюджету між ними

Із цих даних для аналітика витікає наступне – колонка “Вірогідність” пропонує вірогідності походу людини на вакцинацію, перший та другий скринінги. Максимальним значенням є одиниця. Це означає, що людина повинна обов’язково пройти вказану процедуру. Мінімальне значення – нуль, людина не повинна взагалі проходити даний захід.

Колонка “Затрати” вказує на частину бюджету від загально існуючого, яку треба витратити на певну фармакотерапевтичну процедуру. Тобто одиниця значить, що весь бюджет, запланований на певний цикл заходів, потрібно витратити на конкретно цю процедуру. А нуль значить, відповідно, що витрачати бюджет на конкретно цю вакцинацію або діагностичний скринінг – не потрібно.

Відповідні дані аналітик може передавати до медичних закладів із рекомендаціями щодо подальшої стратегії боротьби з хронічними інфекційними захворюваннями серед конкретної популяції.

На вкладці “Графіки” програмного продукту, користувач системи може побудувати графіки залежності (рисунок 5.14). Вони будуються з набору точок, які випадково моделює система.

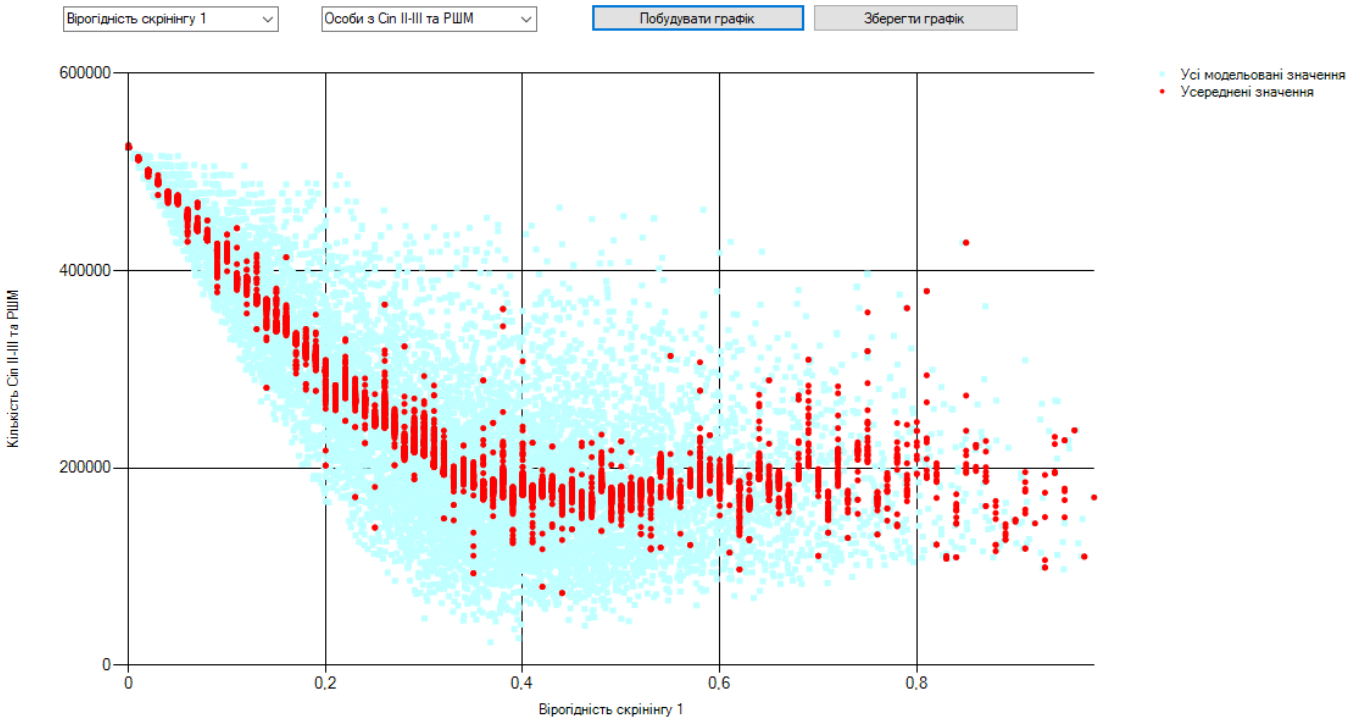


Рисунок 5.14 – Побудова графіків залежності

Блакитні точки на графіку – це усі результати змодельовані системою. Всього відбувається десять тисяч моделювань різних варіантів розвитку подій у популяції.

Червоні точки – середні значення серед усіх моделювань. Вони потрібні для того, щоб аналітик мав можливість розуміти медіану значень та бачити більш наглядні результати, що покращить розуміння залежностей.

За допомогою цих графіків аналітик може зробити певні висновки щодо фармакотерапевтичної стратегії втручання у розвиток хронічних захворювань даної популяції.

Наприклад, для даної популяції, прийнятна кількість людей, що будуть знаходитись у важкому стані дисплазії та матимуть РШМ, знаходиться на рівні від двадцяти до двадцяти п’яти тисяч. За графіком наведеним вище, медичний аналітик може зробити заключення, що для утримання популяційної групи на такому рівні не

потрібно більше, ніж 10-20% вірогідності проходження першого скринінгу. Відповідно до цього будуть прийняті заходи для зменшення чи збільшення залучення людей до цього заходу.

Набір координат точок, відповідно до яких будуть будуватись графіки залежності, можна обрати у двох випадаючих меню (рисунок 5.15). Користувач може отримати тридцять різних графіків використовуючи розроблену інформаційну систему.

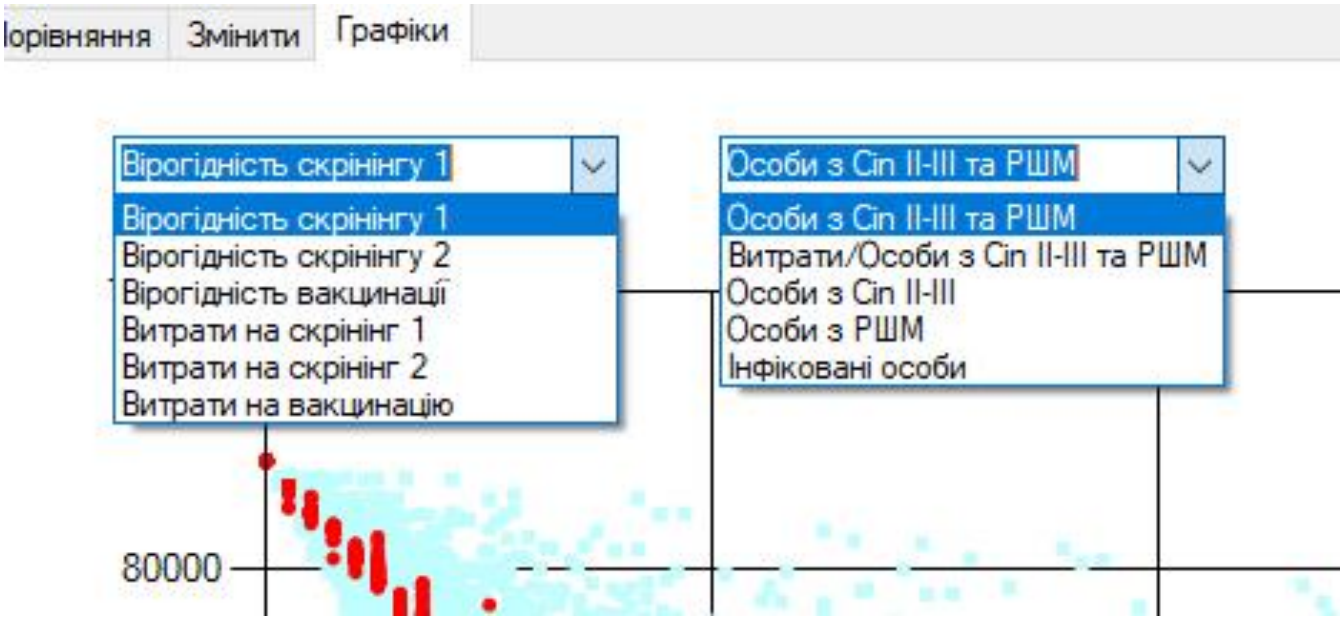


Рисунок 5.15 – Вибір даних, відповідно яким буде побудовано графік залежності

За рахунок цих графіків користувач має змогу побачити такі залежності:

– кількість людей, що знаходяться у стані Cin II-III та РШМ відносно вірогідності першого скринінгу;

– витрати ділені на кількість людей, що знаходяться у стані Cin II-III та РШМ відносно вірогідності першого скринінгу;

– кількість людей, що знаходяться у стані Cin II-III відносно вірогідності першого скринінгу;

– кількість людей, що знаходяться у стані РШМ відносно вірогідності першого скринінгу;

– кількість загальна кількість інфікованих людей відносно вірогідності першого скринінгу;

– кількість людей, що знаходяться у стані Cin II-III та РШМ відносно вірогідності другого скринінгу;

– витрати ділені на кількість людей, що знаходяться у стані Cin II-III та РШМ відносно вірогідності другого скринінгу;

– кількість людей, що знаходяться у стані Cin II-III відносно вірогідності другого скринінгу;

– кількість людей, що знаходяться у стані РШМ відносно вірогідності другого скринінгу;

– кількість загальна кількість інфікованих людей відносно вірогідності другого скринінгу;

– кількість людей, що знаходяться у стані Cin II-III та РШМ відносно вірогідності вакцинації;

– витрати ділені на кількість людей, що знаходяться у стані Cin II-III та РШМ відносно вірогідності вакцинації;

– кількість людей, що знаходяться у стані Cin II-III відносно вірогідності вакцинації;

– кількість людей, що знаходяться у стані РШМ відносно вірогідності вакцинації;

– кількість загальна кількість інфікованих людей відносно вірогідності вакцинації;

– кількість людей, що знаходяться у стані Cin II-III та РШМ відносно витрат на перший скринінг;

– витрати ділені на кількість людей, що знаходяться у стані Cin II-III та РШМ відносно витрат на перший скринінг;

– кількість людей, що знаходяться у стані Cin II-III відносно витрат на перший скринінг;

– кількість людей, що знаходяться у стані РШМ відносно витрат на перший скринінг;

– кількість загальна кількість інфікованих людей відносно витрат на перший скринінг;

– кількість людей, що знаходяться у стані Cin II-III та РШМ відносно витрат на другий скринінг;

– витрати ділені на кількість людей, що знаходяться у стані Cin II-III та РШМ відносно витрат на другий скринінг;

– кількість людей, що знаходяться у стані Cin II-III відносно витрат на другий скринінг;

– кількість людей, що знаходяться у стані РШМ відносно витрат на другий скринінг;

– кількість загальна кількість інфікованих людей відносно витрат на другий скринінг;

– кількість людей, що знаходяться у стані Cin II-III та РШМ відносно витрат на вакцинацію;

– витрати ділені на кількість людей, що знаходяться у стані Cin II-III та РШМ витрат на вакцинацію;

– кількість людей, що знаходяться у стані Cin II-III відносно витрат на вакцинацію;

– кількість людей, що знаходяться у стані РШМ відносно витрат на вакцинацію;

– кількість загальна кількість інфікованих людей відносно витрат на вакцинацію.

Використовуючи інформацію, отриману з цих графіків, аналітика має змогу оцінити можливі варіанти розвитку подій при певних значеннях імовірностей фармакотерапевтичних втручань, а також стратегій витрат на відповідні медичні заходи (вакцинації та діагностичні скринінги). Вони наглядно демонструють кількість людей популяції, які опиняться в певному стані хвороби по кожному з цих значень (від нуля до одиниці).

За рахунок цього аналітик має змогу створити власну стратегію медичного втручання, що може відрізнятися від запропонованої інформаційною системою. Це

може знадобитись, наприклад, для аналізу вже існуючої стратегії, щоб оцінити її наслідки.

Користувач може зберегти собі будь-який зі вказаних графіків окремо, натиснувши кнопку “Зберегти графік” на вкладці “Графіки” (рисунок 5.16).

Рисунок 5.16 – Кнопки, що відповідають за побудову та збереження графіків

Для більш зручного перегляду наслідків та оцінки вже існуючої стратегії фармакотерапевтичних втручань для цієї ж групи людей, існує можливість змоделювати популяцію, відповідно до конкретних значень, які пропонує певний медичний заклад, інститут тощо.

Це можна зробити на вкладці “Змінити” даного програмного додатку. Тут аналітик має можливість встановити значення імовірностей певних медичних заходів (першого та другого діагностичних скринінгів і вакцинації) та відповідні витрати на ці заходи вручну і змоделювати популяцію.

На цій же вкладці користувач інформаційної системи може передивитись результати моделювання популяції. Інформація про популяцію подається у такому ж вигляді, як а на вкладці “Порівняння”. Тобто у таблицях, структурованими за кроками, групами вірусу папіломи людини та віком (рисунок 5.17).

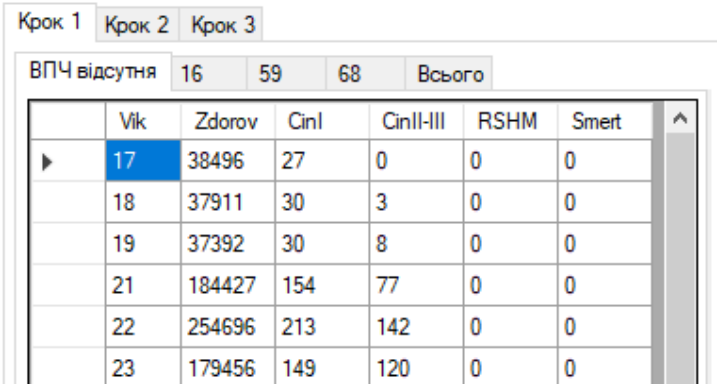


Рисунок 5.17 – Таблиця розвитку популяції з вкладки “Змінити”

Також на вкладці “Змінити” користувач може отримати частково оптимальну стратегію фармакотерапевтичного втручання, за допомогою інформаційної системи. Це можна зробити поставивши галочки навпроти потрібних характеристик (рисунок 5.18). В такому разі користувач вводить вже відомі йому дані та не помічає їх галочками. Дані, які він хоче оптимізувати він відмічає галочкою, тим самим надаючи системі можливість згенерувати оптимальні значення при деяких встановлених.

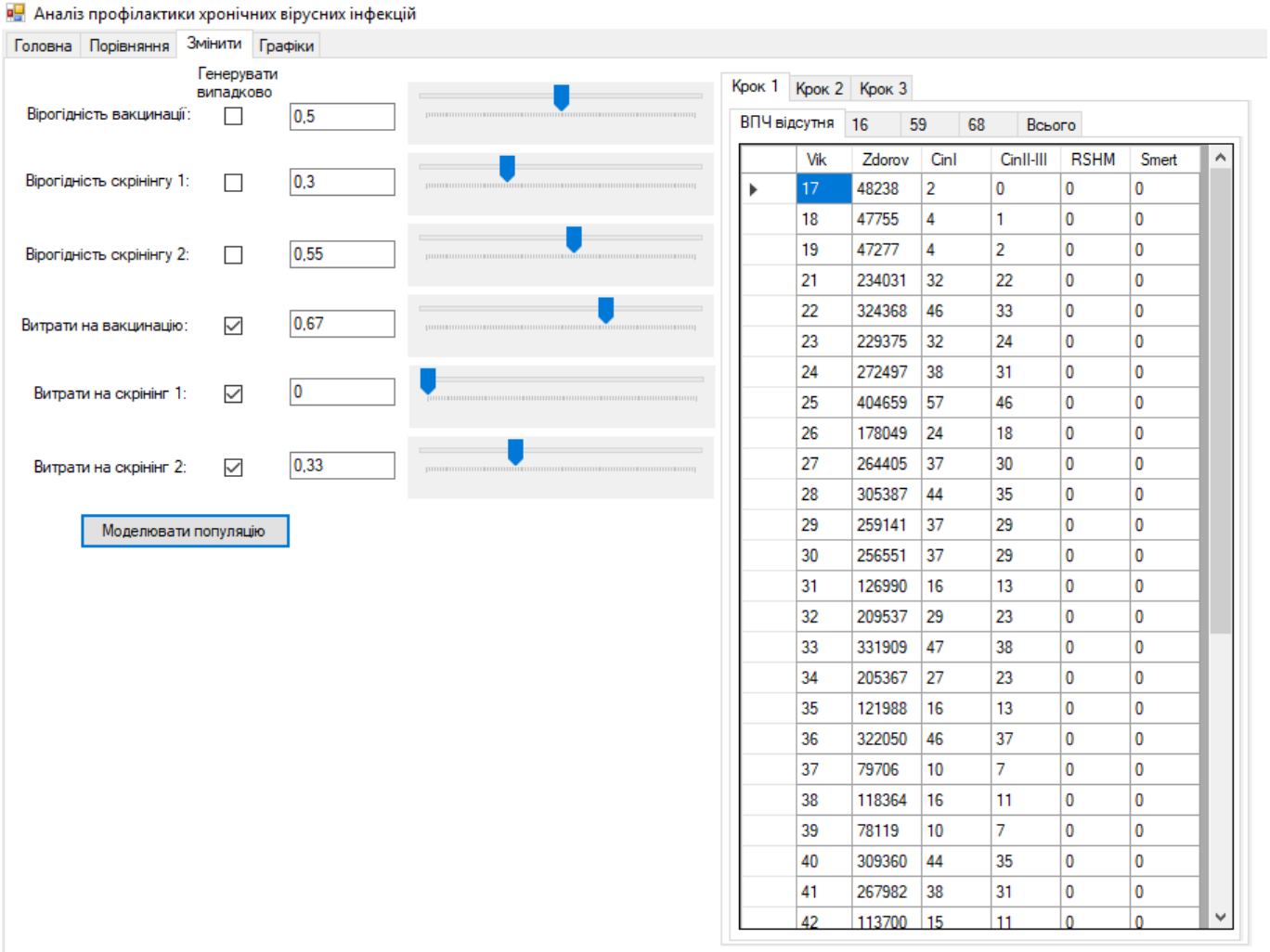


Рисунок 5.18 – Вкладка “Змінити”

Якщо користувачу системи вже відомо певні дані, відносно імовірностей діагностичних скринінгів або вакцинації, а також розподілу бюджету між ними, може задати їх власноруч. Дані можна ввести використовуючи відповідні слайдери

та текстові поля (рисунок 5.19). Після чого користувач зможе побачити значення, яке він ввів використовуючи слайдер, у відповідному текстовому полі, що знаходиться поряд.

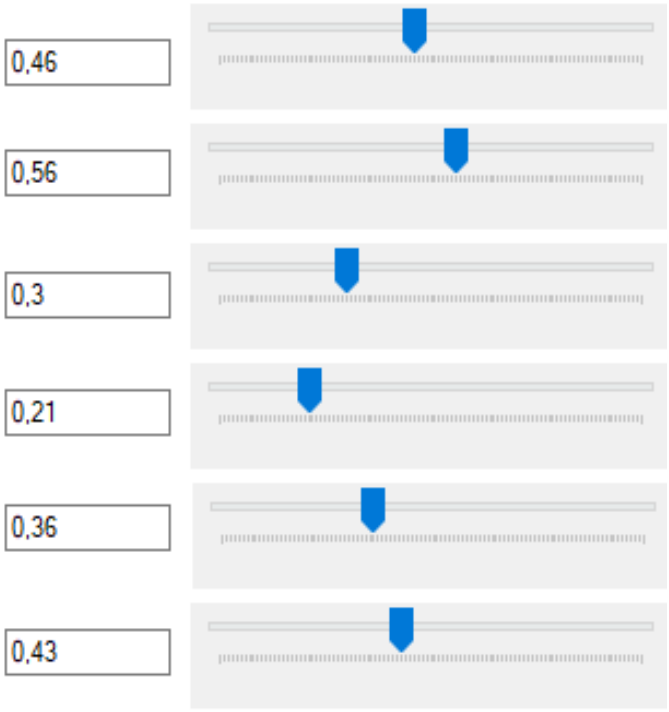


Рисунок 5.19 – Слайдери, що встановлюють відповідні значення імовірностей проходжень фармакотерапевтичних значень та витрат на них

Для більшої точності, в програмному продукті передбачене введення цих даних з клавіатури. Це можна зробити у потрібному текстовому полі, що знаходиться поряд з відповідним слайдером. Точність введених даних обмежена однією сотою від одиниці (або одним процентом).

Також користувачу надається можливість обрати, які значення він хоче отримати від системи, тобто оптимальні значення. Для цього потрібно поставити галочки біля відповідних параметрів (рисунок 5.20). До них відносяться:

– вірогідність вакцинації;

– вірогідність скринінгу 1;

– вірогідність скринінгу 2;

– витрати на вакцинацію;

– витрати на скринінг 1;

– витрати на скринінг 2.

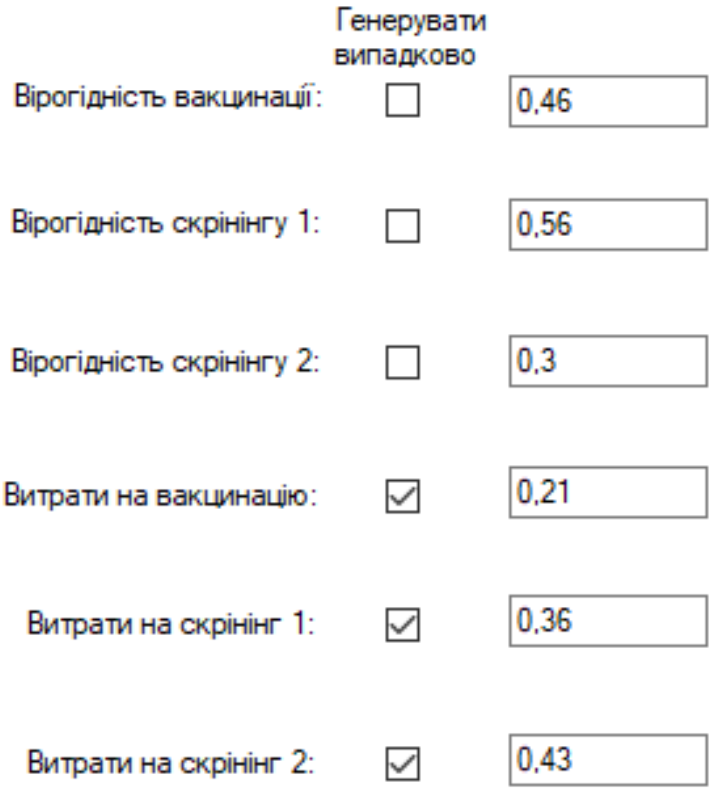


Рисунок 5.20 – Вибір параметрів, для яких значення буде згенеровано системою

При бажанні зберегти усі графіки та стислий звіт у форматі документу Excel користувачу потрібно натиснути кнопку “Зберегти звіт” на вкладці “Головна” (рисунок 5.21).

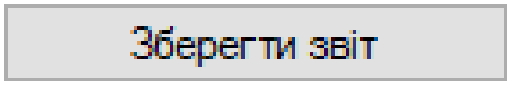


Рисунок 5.21 – Кнопка для збереження звіту

Після чого з’явиться діалогове вікно (рисунок 5.22). В ньому потрібно буде вказати назву документа та шлях, за яким звіт буде збережено. У документі буде міститись інформація про імовірності проходження першого та другого

діагностичних скринінгів, імовірність проходження вакцинації людиною. В звіті користувач також зможе знайти інформацію про загальну кількість пацієнтів, що будуть знаходитись у важкому стані дисплазії або РШМ та загальні витрати. Також у файлі будуть знаходитись картинки з графіками залежності, які генерував користувач.

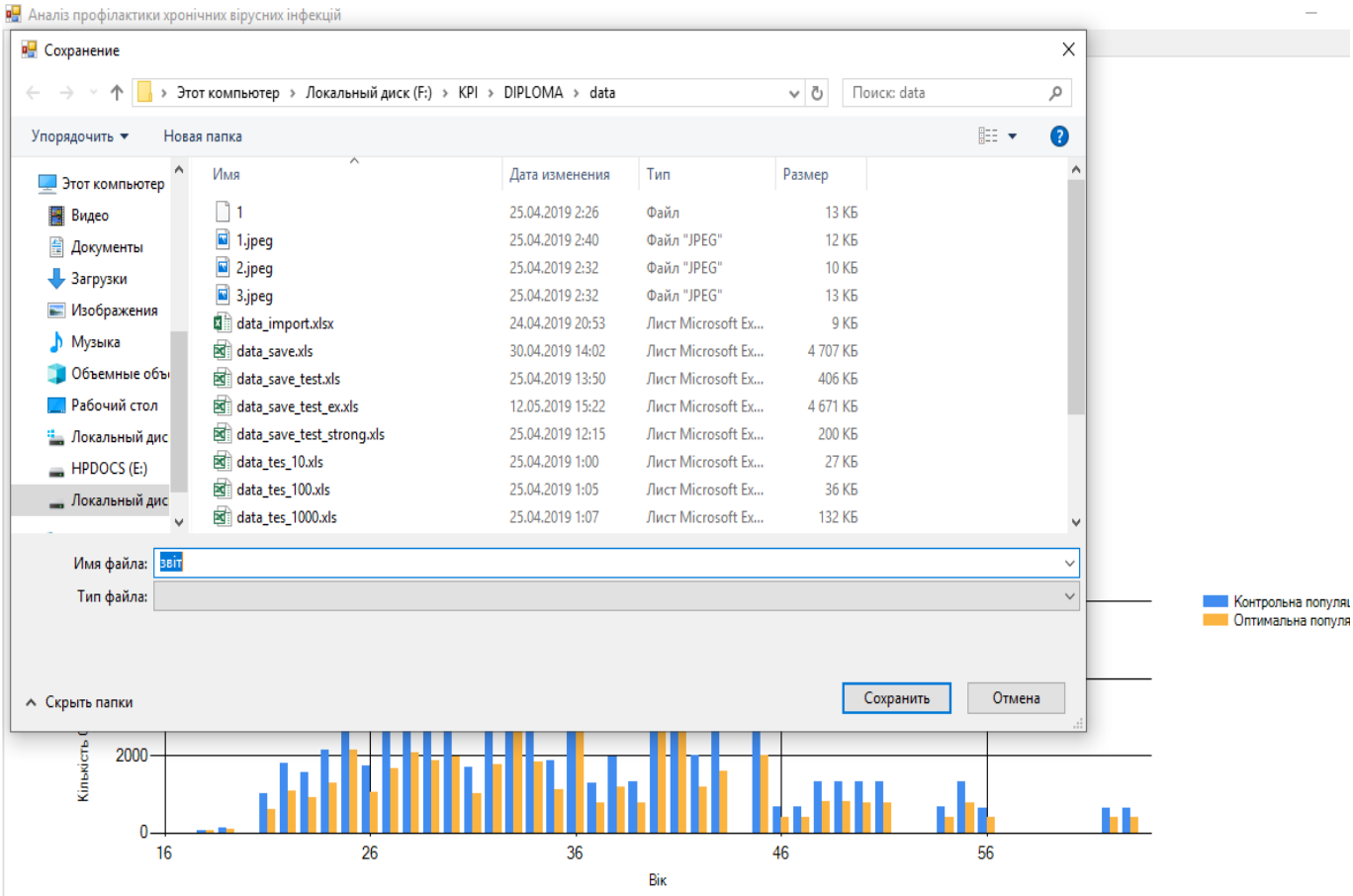


Рисунок 5.22 – Діалогове вікно, що відповідає за збереження звіту

Після збереження звіту користувач може продовжити роботу з програмним додатком. Загрузити нові вхідні дані, змоделювати нові популяції, продовжити роботу зі старими тощо.

В разі, якщо користувач отримав значення, які йому були потрібні, для закінчення роботи з програмою достатньо натиснути червоний хрестик в правому верхньому куті.

Висновки до розділу

У даному розділі було розглянуто розроблений програмний додаток для вирішення поставленої задачі.

Також були описані системні вимоги до комп’ютера користувача, які сприятимуть коректній роботі програмного продукту.

Була описана робота користувача з системою.

ВИСНОВКИ

В ході виконання дипломної роботи було розроблено інформаційну систему для ефективного менеджменту хронічних інфекційних захворювань.

Додаток було написано при використанні об’єктно-орієнтованої мови програмування C# та з використанням графічного інтерфейсу Windows Forms і відповідних бібліотек.

Програму можна використовувати для моделювання великих популяцій людей маючи невелику вхідну вибірку. А також для порівняння ефективності різних стратегій фармакотерапевтичних втручань, які користувач програми може отримувати як вручну (частково або повністю), так і відповідно до запропонованого системою варіанта.

В ході роботи було проведено огляд та зроблено аналіз засобів, які були використані для створення даного програмного забезпечення. До них відносяться середовище розробки Visual Studio 2017, графічний інтерфейс програмування Windows Forms та відповідна бібліотека мови C#, що відповідає за роботу програми з документами створеними у програмі Microsoft Office Excel – Microsoft.Office.Interop.Excel.

Розроблена інформаційна система не потребує спеціального обладнання або надпотужного апаратного забезпечення. Для коректної роботи з даним програмним забезпеченням необхідний лише комп’ютер середньої потужності та гарнітури для роботи користувача з системою.

Користувач має змогу власноруч задавати параметри системи для моделювання популяції людей та отримання оптимального варіанту стратегії втручання у фармакотерапію діагностичних скринінгів та вакцинацій. Після чого він отримує відповідні графіки залежностей певних параметрів, які сам він може обрати.

Також користувачу інформаційної системи надається можливість зберегти звіт із результатами роботи програми у форматі документа Excel (.xlsx) та окремі графіки у форматі картинок зі стандартним розширенням .png.

Результати роботи програмного продукту мають невелику похибку. Використовуючи мінімальну вхідну групу людей, користувач може моделювати популяцію з похибкою не більше 25%. Якщо вхідна популяція буде більшою, то процент похибки зменшується до 5-15%.

Ця програма робить можливим процес прогнозування розвитку популяції з урахуванням хронічних інфекційних захворювань та їх прогресії у певної групи людей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Соловьев С.А. Фармакоэкономический анализ для оценки стратегий диагностики респираторных вирусных инфекций / С.А. Соловьев, Я.А. Дзюблик, О.В. Обертинская, И.В. Дзюблик // Рецепт. – 2014. – №6. – С.119–128.
2. Roland KB, Soman A, ct al. Human papillomavirus and Papanicolaou tests screening interval recommendations in the United States. Am J Obstet 2011; 205; 447; el-8.
3. Price CP. Application of the principles of evidence-based medicine to laboratory medicine. Clinica Chimica Acta. – 2003. – P. 47–54.
4. Health, United States; Hyattsville: National Center for Health Statistics. – 2008.
5. Baltimore: Centers for Medicare and Medicaid. Medicare National Coverage Determinations (NCD) coding policy manual and change report. – January 2009.
6. Harris RP, Helfand M, Woolf SH, et al. Current methods of the U.S. Preventive Services Task Force: a review of the process. Am J Prev Med. – 2001. – P. 20–35.
7. Hethcote H. W. The mathematics of infectious diseases. SIAM Review. – 2000. – P. 599–653.
8. Microsoft Office [Електронний ресурс] – Режим доступу до ресурсу: https://www.office.com/.
9. MathWorks MATLAB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/products/matlab.html.
10. Mathsoft Mathcad [Електронний ресурс] – Режим доступу до ресурсу: https://www.ptc.com/en/products/mathcad/.
11. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/.
12. Windows Forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview/.

13. Qt Documentation [Електронний ресурс] – Режим доступу до ресурсу: http://qt-project.org/doc/.
14. Visual Studio Library [Електронний ресурс] – Режим доступу до ресурсу: http://msdn.microsoft.com/en-us/library/vstudio.
15. Доступ до об’єктів взаємодії Office за допомогою Visual C# [Електронний ресурс] - Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/interop/how-to-access-office-onterop-objects.
16. Microsoft.Office.Interop.Excel Namespace [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.excel?view=excel-pia.
17. System.Drawing Namespace [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing?view=netframework-4.8.
18. Brown E. Windows Forms Programming with C#. Manning. – 2002. – P. 25–49.
19. Schildt H. C# : The Complete Reference. McGraw-Hill Osborne. – 2002. – P. 130–139.
20. Hethcote H. W. The mathematics of infectious diseases. SIAM Review. – 2000. – P. 599–653.
21. Alex F Bielajew Fundamentals of the Monte Carlo method for neutral and charged particle transport. The Unicersity of Michigan. – 2000. – P.15-25.
22. N. Metropolis, S. Ulam, The Monte Carlo Method, J. Amer. statistical assoc. – 1949. – P.335–341.
23. Apostolos N. Burnetas and Michael N. Katehakis, Optimal Adaptive Policies for Markov Decision Processes. Mathematics of Operations Research – 1997. – P.222-255.
24. Филиппенко Н. Г. Методические аспекты клинико-экономического исследования: метод. рекомендации для студентов, ординаторов, аспирантов мед. вузов, врачей и провизоров / Н. Г. Филиппенко, С. В. Поветкин // КГМУ. – Курск: КГМУ. – 2003. – С.20.

25. Соловьев С.А. Фармакоэкономический анализ для оценки стратегий диагностики респираторных вирусных инфекций / С.А. Соловьев, Я.А. Дзюблик, О.В. Обертинская, И.В. Дзюблик // Рецепт. – 2014. – №6. – С.119–128.

ДОДАТОК А

Інформаційна технологія для ефективного менеджменту хронічних захворювань

Специфікація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5274_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5274_19Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5274_19Б	WindowsFormsApp1.exe	Основний компонент інформаційної системи
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5274_19Б	data_import.xlsx	Компонент з вхідними даними

ДОДАТОК Б

Інформаційна технологія для ефективного менеджменту хронічних захворювань

Текст програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5274_19Б

Аркушів 10

Київ 2019

Текст підпрограми калібрування системи

```
private void button1_Click(object sender, EventArgs e)
    {
      if (btn_file_open_pressed == true)
      {
        btn_calibrate_pressed = true;
        double matr_max1 = 1, matr_max2 = 1, matr_max3 = 1, matr_max4 = 1, matr_max5 = 1;
        Random rand = new Random();
        for (int t = 0; t < 38; t++)
        {
          dataGridView29.Rows[t].Cells[0].Value = dataGridView30.Rows[t].Cells[0].Value = population[t, 0];
        }
        for (int i = 0; i < 38; i++)
        {
          dataGridView1.Rows[i].Cells[0].Value = population[i, 0];
          dataGridView1.Rows[i].Cells[1].Value = population[i, 1];
          dataGridView1.Rows[i].Cells[2].Value = population[i, 2];
          dataGridView1.Rows[i].Cells[3].Value = population[i, 3];
          dataGridView1.Rows[i].Cells[4].Value = population[i, 4];
          dataGridView1.Rows[i].Cells[5].Value = population[i, 5];
          dataGridView1.Rows[i].Cells[6].Value = population[i, 6];
          dataGridView1.Rows[i].Cells[7].Value = population[i, 7];
        }
        for (int i = 0; i < 38; i++)
        {
          part_sum = population[i, 1] + population[i, 2] + population[i, 3];
          part_rshm = 1 - part_sum;
          population_part[i, 0] = population[i, 1] / part_sum;
          population_part[i, 1] = population[i, 2] / part_sum;
          population_part[i, 2] = population[i, 3] / part_sum;
          vpch_part_sum = population[i, 5] + population[i, 6] + population[i, 7] + 0.0000000001;
          population_part[i, 5] = population[i, 5] / vpch_part_sum;
          population_part[i, 6] = population[i, 6] / vpch_part_sum;
          population_part[i, 7] = population[i, 7] / vpch_part_sum;
        }
        for (int tr = 0; tr < 100; tr++)
        {
          matr_max1 = 1; matr_max2 = 1; matr_max3 = 1; matr_max4 = 1; matr_max5 = 1;
          pohybka = 0;
          matr_rand_1[0] = Convert.ToDouble(rand.Next(0, 100) / 100.0);
          matr_rand_1[1] = 1 - matr_rand_1[0];
          matr_rand_1[2] = 0;
          matr_rand_1[3] = 0;
          matr_rand_1[4] = 0;
          while (matr_rand_2[2] <= 0 || matr_rand_2[2] >= 1)
          {
            matr_rand_2[0] = Convert.ToDouble(rand.Next(0, 100) / 100.0);
            matr_rand_2[1] = Convert.ToDouble(rand.Next(Convert.ToInt32(1 - matr_rand_2[0]), 100) / 100.0);
            matr_rand_2[2] = 1 - matr_rand_2[1] - matr_rand_2[0];
          }
          matr_rand_2[3] = 0;
          matr_rand_2[4] = 0;
          for (int r3 = 2; r3 < 3; r3++)
          {
            matr_rand_3[r3] = Convert.ToDouble(rand.Next(0, 100) / 100.0);
            matr_max3 -= matr_rand_3[r3];
          }
          matr_rand_3[3] = matr_max3;
          matr_rand_3[4] = 0;
          matr_rand_4[0] = 0;
          matr_rand_4[1] = 0;
          matr_rand_4[2] = 0;
          matr_rand_4[3] = Convert.ToDouble(rand.Next(0, 100) / 100.0);
          matr_rand_4[4] = 1 - matr_rand_4[3];
          matr_rand_5[0] = 0;
          matr_rand_5[1] = 0;
          matr_rand_5[2] = 0;
          matr_rand_5[3] = 0;
          matr_rand_5[4] = 1;
          population_part_new[0, 0] = matr_rand_1[0] * population[0, 1] + matr_rand_2[0] * population[0, 2] + matr_rand_3[0] * population[0, 3] +
matr_rand_4[0] * population[0, 4] + matr_rand_5[0] * population[0, 4];
          population_part_new[0, 1] = matr_rand_1[1] * population[0, 1] + matr_rand_2[1] * population[0, 2] + matr_rand_3[1] * population[0, 3] +
matr_rand_4[1] * population[0, 4] + matr_rand_5[1] * population[0, 4];
          population_part_new[0, 2] = matr_rand_1[2] * population[0, 1] + matr_rand_2[2] * population[0, 2] + matr_rand_3[2] * population[0, 3] +
matr_rand_4[2] * population[0, 4] + matr_rand_5[2] * population[0, 4];
          population_part_new[0, 3] = matr_rand_1[3] * population[0, 1] + matr_rand_2[3] * population[0, 2] + matr_rand_3[3] * population[0, 3] +
matr_rand_4[3] * population[0, 4] + matr_rand_5[3] * population[0, 4];
```

```
            population_part_new[0, 4] = matr_rand_1[4] * population[0, 1] + matr_rand_2[4] * population[0, 2] + matr_rand_3[4] * population[0, 3] +
matr_rand_4[4] * population[0, 4] + matr_rand_5[4] * population[0, 4];
            for (int j = 1; j < 38; j++)
            {
               population_part_new[j, 0] = matr_rand_1[0] * population_part_new[j - 1, 0] + matr_rand_2[0] * population_part_new[j - 1, 1] +
matr_rand_3[0] * population_part_new[j - 1, 2] + matr_rand_4[0] * population_part_new[j - 1, 3] + matr_rand_5[0] * population_part_new[j - 1, 4];
               population_part_new[j, 1] = matr_rand_1[1] * population_part_new[j - 1, 0] + matr_rand_2[1] * population_part_new[j - 1, 1] +
matr_rand_3[1] * population_part_new[j - 1, 2] + matr_rand_4[1] * population_part_new[j - 1, 3] + matr_rand_5[1] * population_part_new[j - 1, 4];
               population_part_new[j, 2] = matr_rand_1[2] * population_part_new[j - 1, 0] + matr_rand_2[2] * population_part_new[j - 1, 1] +
matr_rand_3[2] * population_part_new[j - 1, 2] + matr_rand_4[2] * population_part_new[j - 1, 3] + matr_rand_5[2] * population_part_new[j - 1, 4];
               population_part_new[j, 3] = matr_rand_1[3] * population_part_new[j - 1, 0] + matr_rand_2[3] * population_part_new[j - 1, 1] +
matr_rand_3[3] * population_part_new[j - 1, 2] + matr_rand_4[3] * population_part_new[j - 1, 3] + matr_rand_5[3] * population_part_new[j - 1, 4];
               population_part_new[j, 4] = matr_rand_1[4] * population_part_new[j - 1, 0] + matr_rand_2[4] * population_part_new[j - 1, 1] +
matr_rand_3[4] * population_part_new[j - 1, 2] + matr_rand_4[4] * population_part_new[j - 1, 3] + matr_rand_5[4] * population_part_new[j - 1, 4];
               pohybka += Math.Abs(population_part[j, 0] - population_part_new[j, 0]) + Math.Abs(population_part[j, 1] - population_part_new[j, 1]) +
Math.Abs(population_part[j, 2] - population_part_new[j, 2])
                  + Math.Abs(population_part[j, 3] - population_part_new[j, 3]) + Math.Abs(population_part[j, 4] - population_part_new[j, 4]);
            }
            if (pohybka < pohybka_start)
            {
               for (int ph = 0; ph < 38; ph++)
               {
                  best_population_part[ph, 0] = population_part_new[ph, 0];
                  best_population_part[ph, 1] = population_part_new[ph, 1];
                  best_population_part[ph, 2] = population_part_new[ph, 2];
                  best_population_part[ph, 3] = population_part_new[ph, 3];
                  best_population_part[ph, 4] = population_part_new[ph, 4];
               }
               for (int ph_ver = 0; ph_ver < 5; ph_ver++)
               {
                  best_matr_rand_1[ph_ver] = matr_rand_1[ph_ver];
                  best_matr_rand_2[ph_ver] = matr_rand_2[ph_ver];
                  best_matr_rand_3[ph_ver] = matr_rand_3[ph_ver];
                  best_matr_rand_4[ph_ver] = matr_rand_4[ph_ver];
                  best_matr_rand_5[ph_ver] = matr_rand_5[ph_ver];
               }
               pohybka_start = pohybka;
            }
}
            double sum_hvor = 0, sum_zdr = 0;
            for (int i = 0; i < 38; i++)
            {
               sum_hvor = population[i, 1] + population[i, 2] + population[i, 3] + population[i, 4] + sum_hvor;
               sum_zdr = population[i, 0] + sum_zdr;
               sum_agent = sum_hvor + sum_zdr;
               poshyrenist_stat = sum_hvor / sum_agent;
            }
            double sum = 0, sum_rshm = 0;
            for (int i = 0; i < 38; i++)
            {
               sum_rshm = best_population_part[i, 3] + sum_rshm;
               sum = best_population_part[i, 0] + best_population_part[i, 1] + best_population_part[i, 2] + best_population_part[i, 3] + best_population_part[i,
4] + sum;
               smertnist_stat = sum_rshm / sum;
            }
            zd_zd = Convert.ToDouble(dataGridView7.Rows[0].Cells[0].Value);
            zd_cinI = Convert.ToDouble(dataGridView7.Rows[1].Cells[0].Value);
            cinI_zd = Convert.ToDouble(dataGridView7.Rows[0].Cells[1].Value);
            cinI_cinI = Convert.ToDouble(dataGridView7.Rows[1].Cells[1].Value);
            cinI_cinII = Convert.ToDouble(dataGridView7.Rows[2].Cells[1].Value);
            cinII_cinII = Convert.ToDouble(dataGridView7.Rows[2].Cells[2].Value);
            cinII_rshm = Convert.ToDouble(dataGridView7.Rows[3].Cells[2].Value);
            rshm_rshm = Convert.ToDouble(dataGridView7.Rows[3].Cells[3].Value);
            rshm_smert = Convert.ToDouble(dataGridView7.Rows[4].Cells[3].Value);
            smert_smert = Convert.ToDouble(dataGridView7.Rows[4].Cells[4].Value);
         }
         else
         {
            MessageBox.Show(@"Спочатку відкрийте файл з вхідними даними.", "", MessageBoxButtons.OK);
         }
      }
```

Текст підпрограми моделювання системи

```
            btn_model_pressed = true;

            agents_total = int.Parse(textBox3.Text);

            ver_vakcinacii = 0;
```

```
            ver_skrining1 = 0;
            ver_skrining_nayavn = 0;
            ver_skrining_vids = 0;
            ver_vpl_prys_disp_nayavn = vpl_agents_sum / age_total;
            ver_vpl_vids_disp_nema = 1 - ver_vpl_prys_disp_nayavn;
            ver_nadannya_med_dopomogi = 0;
            ver_povtorn_skr_1 = 0;
            proshli_skr1 = Math.Truncate(agents_total * ver_skrining1); //vot tut nujen counter vitraty
            neproshli_skr1 = agents_total - proshli_skr1;
            //displas_nayavni_nevidom = Math.Truncate(neproshli_skr1 * poshyrenist_stat); //vot tut nujen counter red
            displas_nayavni_nevidom = Math.Truncate(neproshli_skr1 * zd_cinI); //vot tut nujen counter red
            counter_red += displas_nayavni_nevidom;
            displas_vidsutni_nevidom = neproshli_skr1 - displas_nayavni_nevidom;
            //displas_nayavni_vidom = Math.Truncate(proshli_skr1 * poshyrenist_stat); //vot tut nujen counter red
            displas_nayavni_vidom = Math.Truncate(proshli_skr1 * zd_cinI); //vot tut nujen counter red
            counter_red += displas_nayavni_vidom;
            displas_vidsutni_vidom = proshli_skr1 - displas_nayavni_vidom;
            proshli_skr2_nayavni = Math.Truncate(displas_nayavni_vidom * ver_skrining_nayavn); //vot tut nujen counter vitraty
            neproshli_skr2_nayavni = displas_nayavni_vidom - proshli_skr2_nayavni;
            proshli_skr2_vidsutni = Math.Truncate(displas_vidsutni_vidom * ver_skrining_vids); //vot tut nujen counter vitraty
            neproshli_skr2_vidsutni = displas_vidsutni_vidom - proshli_skr2_vidsutni;
            vpl_prys_disp_nayavn = Math.Truncate(proshli_skr2_nayavni * ver_vpl_prys_disp_nayavn); //vot tut nujen counter red
            counter_red += vpl_prys_disp_nayavn;
            vpl_vids_disp_nayavn = proshli_skr2_nayavni - vpl_prys_disp_nayavn;//vot tut nujen counter red
            counter_red += vpl_vids_disp_nayavn;
            vpl_vids_disp_nema = Math.Truncate(proshli_skr2_vidsutni * ver_vpl_vids_disp_nema);
            vpl_prys_disp_nema = proshli_skr2_vidsutni - vpl_vids_disp_nema;
            prohodyt_med_dopomog_vpl_prys_disp_nayavn = Math.Truncate(vpl_prys_disp_nayavn * ver_nadannya_med_dopomogi);//vot tut nujen counter
vitraty
            prohodyt_med_dopomog_vpl_vids_disp_nayavn = Math.Truncate(vpl_vids_disp_nayavn * ver_nadannya_med_dopomogi); //vot tut nujen counter
vitraty
            povtorn_skr_1 = Math.Truncate(vpl_prys_disp_nayavn * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nayavn * ver_povtorn_skr_1) +
Math.Truncate(vpl_prys_disp_nema * ver_povtorn_skr_1)
                + Math.Truncate(vpl_vids_disp_nema * ver_povtorn_skr_1); //vot tut nujen counter vitra           povtorn_skr_1_disp_vids =
Math.Truncate(vpl_vids_disp_n)
            vpl_prys_disp_nayavn_stay = vpl_prys_disp_nayavn - prohodyt_med_dopomog_vpl_prys_disp_nayavn - Math.Truncate(vpl_prys_disp_nayavn *
ver_povtorn_skr_1);
            vpl_prys_disp_nema_stay = vpl_prys_disp_nema - Math.Truncate(vpl_prys_disp_nema * ver_povtorn_skr_1);
            vpl_vids_disp_nayavn_stay = vpl_vids_disp_nayavn - prohodyt_med_dopomog_vpl_vids_disp_nayavn - Math.Truncate(vpl_vids_disp_nayavn *
ver_povtorn_skr_1);
            vpl_vids_disp_nema_stay = vpl_vids_disp_nema - Math.Truncate(vpl_vids_disp_nema * ver_povtorn_skr_1);
            //for staying
            ////displas_nayavni_nevidom
            //displas_nayavni_nevidom_skr1_step1 = Math.Truncate(displas_nayavni_nevidom * ver_skrining1) + Math.Truncate(vpl_prys_disp_nayavn *
ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nayavn * ver_povtorn_skr_1); //go next
            displas_nayavni_nevidom_skr1_step1 = Math.Truncate(displas_nayavni_nevidom * ver_skrining1) +
Math.Truncate((Math.Truncate(vpl_prys_disp_nayavn * ver_povtorn_skr_1)
                + Math.Truncate(vpl_vids_disp_nayavn * ver_povtorn_skr_1)) * ver_skrining1);
            displas_nayavni_nevidom_zalysh_step1 = displas_nayavni_nevidom + Math.Truncate(vpl_prys_disp_nayavn * ver_povtorn_skr_1) +
Math.Truncate(vpl_vids_disp_nayavn * ver_povtorn_skr_1) - displas_nayavni_nevidom_skr1_step1;
            displas_nayavni_nevidom_zd_step1 = Math.Truncate(displas_nayavni_nevidom_zalysh_step1 * cinI_zd);
            displas_nayavni_nevidom_cinI_step1 = Math.Truncate(displas_nayavni_nevidom_zalysh_step1 * cinI_cinI);
```

```
displas_nayavni_nevidom_cinII_step1 = Math.Truncate(displas_nayavni_nevidom_zalysh_step1 * cinI_cinII);
////displas_vidsutni_nevidom
//displas_vidsutni_nevidom_skr1_step1 = Math.Truncate(displas_vidsutni_nevidom * ver_skrining1) + Math.Truncate(vpl_prys_disp_nema * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nema * ver_povtorn_skr_1);//go next
displas_vidsutni_nevidom_skr1_step1 = Math.Truncate(displas_vidsutni_nevidom * ver_skrining1) + Math.Truncate((Math.Truncate(vpl_prys_disp_nema * ver_povtorn_skr_1)
    + Math.Truncate(vpl_vids_disp_nema * ver_povtorn_skr_1)) * ver_skrining1);
displas_vidsutni_nevidom_zalysh_step1 = displas_vidsutni_nevidom + Math.Truncate(vpl_prys_disp_nema * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nema * ver_povtorn_skr_1) - displas_vidsutni_nevidom_skr1_step1;
displas_vidsutni_nevidom_cinI_step1 = Math.Truncate(displas_vidsutni_nevidom_zalysh_step1 * zd_cinI);
displas_vidsutni_nevidom_zd_step1 = displas_vidsutni_nevidom_zalysh_step1 - displas_vidsutni_nevidom_cinI_step1;
////displas_nayavni_nevidom skr 1
displas_nayavni_nayavni_vidom_step1 = Math.Truncate(displas_nayavni_nevidom_skr1_step1 * zd_cinI); //nayavni
displas_nayavni_vidsutni_vidom_step1 = displas_nayavni_nevidom_skr1_step1 - displas_nayavni_nayavni_vidom_step1; //vidsutni
////displas_vidsutni_nevidom skr 1
displas_vidsutni_nayavni_vidom_step1 += Math.Truncate(displas_vidsutni_nevidom_skr1_step1 * zd_cinI); //nayavni
displas_vidsutni_vidsutni_vidom_step1 = displas_vidsutni_nevidom_skr1_step1 - displas_vidsutni_nayavni_vidom_step1; //vidsutni
////neproshli skrin 2 _ step 0 perehodi vnutri po step 0
displas_nayavni_vidom_skr2_zd_step1 = Math.Truncate(neproshli_skr2_nayavni * cinI_zd);
displas_nayavni_vidom_skr2_cinI_step1 = Math.Truncate(neproshli_skr2_nayavni * cinI_cinI);
displas_nayavni_vidom_skr2_cinII_step1 = Math.Truncate(neproshli_skr2_nayavni * cinI_cinII);
displas_vidsutni_vidom_skr2_zd_step1 = Math.Truncate(neproshli_skr2_vidsutni * zd_zd);
displas_vidsutni_vidom_skr2_cinI_step1 = Math.Truncate(neproshli_skr2_vidsutni * zd_cinI);
////sum and skr2
displas_vidsutni_vidom_step1 = displas_nayavni_vidsutni_vidom_step1 + displas_vidsutni_vidsutni_vidom_step1 + neproshli_skr2_vidsutni; //for next step
displas_nayavni_vidom_step1 = displas_nayavni_nayavni_vidom_step1 + displas_vidsutni_nayavni_vidom_step1 + neproshli_skr2_nayavni; //for next step

displas_vidsutni_vidom_skr2_step1 = Math.Truncate(displas_vidsutni_vidom_step1 * ver_skrining_vids);//go next
displas_vidsutni_vidom_skr2_zalysh_step1 = displas_vidsutni_vidom_step1 - displas_vidsutni_vidom_skr2_step1;
displas_nayavni_vidom_skr2_step1 = Math.Truncate(displas_nayavni_vidom_step1 * ver_skrining_nayavn);//go next
displas_nayavni_vidom_skr2_zalysh_step1 = displas_nayavni_vidom_step1 - displas_nayavni_vidom_skr2_step1;
////displas_nayavni_vidom_skr2_step1
vpl_prys_disp_nayavn_step1 = Math.Truncate(displas_nayavni_vidom_skr2_step1 * ver_vpl_prys_disp_nayavn);
vpl_vids_disp_nayavn_step1 = displas_nayavni_vidom_skr2_step1 - vpl_prys_disp_nayavn_step1;
////displas_vids_vidom_skr2_step1
vpl_vids_disp_nema_step1 = Math.Truncate(displas_vidsutni_vidom_skr2_step1 * ver_vpl_vids_disp_nema);
vpl_prys_disp_nema_step1 = displas_vidsutni_vidom_skr2_step1 - vpl_vids_disp_nema_step1;
////med dopomoga
prohodyt_med_dopomog_vpl_prys_disp_nayavn_step1 = Math.Truncate(vpl_prys_disp_nayavn_step1 * ver_nadannya_med_dopomogi);//vot tut nujen counter vitraty
prohodyt_med_dopomog_vpl_vids_disp_nayavn_step1 = Math.Truncate(vpl_vids_disp_nayavn_step1 * ver_nadannya_med_dopomogi); //vot tut nujen counter vitraty
////provrorn skr1 save
povtorn_skr_1_disp_nayavn_step1 = Math.Truncate(vpl_prys_disp_nayavn_step1 * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nayavn_step1 * ver_povtorn_skr_1) + Math.Truncate(vpl_prys_disp_nema_step1 * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nema_step1 * ver_povtorn_skr_1); //vot tut nujen counter vitraty
////stay going next step
vpl_prys_disp_nayavn_stay_step1 = vpl_prys_disp_nayavn_step1 - prohodyt_med_dopomog_vpl_prys_disp_nayavn_step1 - Math.Truncate(vpl_prys_disp_nayavn_step1 * ver_povtorn_skr_1);
vpl_prys_disp_nema_stay_step1 = vpl_prys_disp_nema_step1 - Math.Truncate(vpl_prys_disp_nema_step1 * ver_povtorn_skr_1);
```

```
            vpl_vids_disp_nayavn_stay_step1 = vpl_vids_disp_nayavn_step1 - prohodyt_med_dopomog_vpl_vids_disp_nayavn_step1 -
Math.Truncate(vpl_vids_disp_nayavn_step1 * ver_povtorn_skr_1);
            vpl_vids_disp_nema_stay_step1 = vpl_vids_disp_nema_step1 - Math.Truncate(vpl_vids_disp_nema_step1 * ver_povtorn_skr_1);
            vpl_prys_disp_nayavn_stay_step1_zd = Math.Truncate(vpl_prys_disp_nayavn_stay * cinI_zd);
            vpl_prys_disp_nayavn_stay_step1_cinI = Math.Truncate(vpl_prys_disp_nayavn_stay * cinI_zd);
            vpl_prys_disp_nayavn_stay_step1_cinII = Math.Truncate(vpl_prys_disp_nayavn_stay * cinI_zd);
            vpl_prys_disp_nema_stay_step1_zd = Math.Truncate(vpl_prys_disp_nema_stay * zd_zd_l_vp);
            vpl_prys_disp_nema_stay_step1_cinI = Math.Truncate(vpl_prys_disp_nema_stay * zd_cinI_l_vp);
            vpl_vids_disp_nayavn_stay_step1_zd = Math.Truncate(vpl_vids_disp_nayavn * cinI_zd_l_vv);
            vpl_vids_disp_nayavn_stay_step1_cinI = Math.Truncate(vpl_vids_disp_nayavn * cinI_cinI_l_vv);
            vpl_vids_disp_nayavn_stay_step1_cinII = Math.Truncate(vpl_vids_disp_nayavn * cinI_cinII_l_vv);
            vpl_vids_disp_nema_stay_step1_zd = Math.Truncate(vpl_vids_disp_nema_stay * zd_zd_bl_vv);
            vpl_vids_disp_nema_stay_step1_cinI = Math.Truncate(vpl_vids_disp_nema_stay * zd_cinI_bl_vv);
            povtorn_skr_1_step1 = Math.Truncate(vpl_prys_disp_nayavn_step1 * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nayavn_step1 *
ver_povtorn_skr_1)
                + Math.Truncate(vpl_prys_disp_nema_step1 * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nema_step1 * ver_povtorn_skr_1);
            double[] yValues = {(displas_nayavni_nevidom_zd_step1 + displas_vidsutni_nevidom_zd_step1 + displas_nayavni_vidom_skr2_zd_step1 +
displas_vidsutni_vidom_skr2_zd_step1 + vpl_prys_disp_nayavn_stay_step1_zd + vpl_prys_disp_nema_stay_step1_zd +
                vpl_vids_disp_nayavn_stay_step1_zd + vpl_vids_disp_nema_stay_step1_zd), (displas_nayavni_nevidom_cinI_step1 +
displas_vidsutni_nevidom_cinI_step1 + displas_nayavni_vidom_skr2_cinI_step1 + displas_vidsutni_vidom_skr2_cinI_step1 +
vpl_prys_disp_nayavn_stay_step1_cinI
                + vpl_vids_disp_nayavn_stay_step1_cinI + vpl_vids_disp_nema_stay_step1_cinI), (displas_nayavni_nevidom_cinII_step1 +
displas_nayavni_vidom_skr2_cinII_step1 + vpl_prys_disp_nayavn_stay_step1_cinII + vpl_vids_disp_nayavn_stay_step1_cinII), 0, 0,
            (displas_nayavni_nevidom_zalysh_step1 + displas_vidsutni_nevidom_zalysh_step1)};
            string[] xValues = { "Здорові", "Сін I", "Сін II-III", "РШМ", "Померлі", "Невідомі" };
            for (int s1 = 0; s1 < 38; s1++)
            {
                dataGridView9.Rows[s1].Cells[1].Value = Math.Truncate((displas_nayavni_nevidom_zd_step1 + displas_vidsutni_nevidom_zd_step1 +
displas_nayavni_vidom_skr2_zd_step1 + displas_vidsutni_vidom_skr2_zd_step1 + vpl_prys_disp_nayavn_stay_step1_zd +
vpl_prys_disp_nema_stay_step1_zd +
                vpl_vids_disp_nayavn_stay_step1_zd + vpl_vids_disp_nema_stay_step1_zd) * age_part[s1] * best_population_part[s1, 0]);
                dataGridView9.Rows[s1].Cells[2].Value = Math.Truncate((displas_nayavni_nevidom_cinI_step1 + displas_vidsutni_nevidom_cinI_step1 +
displas_nayavni_vidom_skr2_cinI_step1 + displas_vidsutni_vidom_skr2_cinI_step1 + vpl_prys_disp_nayavn_stay_step1_cinI
                + vpl_vids_disp_nayavn_stay_step1_cinI + vpl_vids_disp_nema_stay_step1_cinI) * age_part[s1] * best_population_part[s1, 1]);
                dataGridView9.Rows[s1].Cells[3].Value = Math.Truncate((displas_nayavni_nevidom_cinII_step1 + displas_nayavni_vidom_skr2_cinII_step1 +
vpl_prys_disp_nayavn_stay_step1_cinII + vpl_vids_disp_nayavn_stay_step1_cinII) * age_part[s1]
                    * best_population_part[s1, 2]);
            }
            ////displas_nayavni_nevidom
            displas_nayavni_nevidom_skr1_step2 = Math.Truncate(displas_nayavni_nevidom_zalysh_step1 * ver_skrining1) +
Math.Truncate((Math.Truncate(vpl_prys_disp_nayavn_step1 * ver_povtorn_skr_1)
                + Math.Truncate(vpl_vids_disp_nayavn_step1 * ver_povtorn_skr_1)) * ver_skrining1);
            displas_nayavni_nevidom_zalysh_step2 = displas_nayavni_nevidom_zalysh_step1 + Math.Truncate(vpl_prys_disp_nayavn_step1 *
ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nayavn_step1 * ver_povtorn_skr_1) - displas_nayavni_nevidom_skr1_step2;
            displas_nayavni_nevidom_zd_step2 = Math.Truncate(displas_nayavni_nevidom_zd_step1 * zd_zd) +
Math.Truncate(displas_nayavni_nevidom_cinI_step1 * cinI_zd);
            displas_nayavni_nevidom_cinI_step2 = Math.Truncate(displas_nayavni_nevidom_cinI_step1 * cinI_cinI) +
Math.Truncate(displas_nayavni_nevidom_zd_step1 * zd_cinI);
            displas_nayavni_nevidom_cinII_step2 = Math.Truncate(displas_nayavni_nevidom_cinII_step1 * cinII_cinII) +
Math.Truncate(displas_nayavni_nevidom_cinI_step1 * cinI_cinII);
            displas_nayavni_nevidom_rshm_step2 = Math.Truncate(displas_nayavni_nevidom_cinII_step1 * cinII_rshm);
            ////displas_vidsutni_nevidom
```

```
displas_vidsutni_nevidom_skr1_step2 = Math.Truncate(displas_vidsutni_nevidom_zalysh_step1 * ver_skrining1) +
Math.Truncate((Math.Truncate(vpl_prys_disp_nema_step1 * ver_povtorn_skr_1)
    + Math.Truncate(vpl_vids_disp_nema_step1 * ver_povtorn_skr_1)) * ver_skrining1);
displas_vidsutni_nevidom_zalysh_step2 = displas_vidsutni_nevidom_zalysh_step1 + Math.Truncate(vpl_prys_disp_nema_step1 *
ver_povtorn_skr_1)
    + Math.Truncate(vpl_vids_disp_nema_step1 * ver_povtorn_skr_1) - displas_vidsutni_nevidom_skr1_step2;
displas_vidsutni_nevidom_zd_step2 = Math.Truncate(displas_vidsutni_nevidom_zd_step1 * zd_zd) +
Math.Truncate(displas_vidsutni_nevidom_cinI_step1 * cinI_zd);
displas_vidsutni_nevidom_cinI_step2 = Math.Truncate(displas_vidsutni_nevidom_cinI_step1 * cinI_cinI) +
Math.Truncate(displas_vidsutni_nevidom_zd_step1 * zd_cinI);
displas_vidsutni_nevidom_cinII_step2 = Math.Truncate(displas_vidsutni_nevidom_cinI_step1 * cinII_cinII);
////displas_nayavni_nevidom skr 1
displas_nayavni_nayavni_vidom_step2 = Math.Truncate(displas_nayavni_nevidom_skr1_step2 * zd_cinI); //nayavni
displas_nayavni_vidsutni_vidom_step2 = displas_nayavni_nevidom_skr1_step2 - displas_nayavni_nayavni_vidom_step2; //vidsutni
////displas_vidsutni_nevidom skr 1
displas_vidsutni_nayavni_vidom_step2 += Math.Truncate(displas_vidsutni_nevidom_skr1_step2 * zd_cinI); //nayavni
displas_vidsutni_vidsutni_vidom_step2 = displas_vidsutni_nevidom_skr1_step2 - displas_vidsutni_nayavni_vidom_step2; //vidsutni
////neproshli skrin 2 _ step 0 perehodi vnutri po step 0
displas_nayavni_vidom_skr2_zd_step2 = Math.Truncate(displas_nayavni_vidom_skr2_cinI_step1 * cinI_zd) +
Math.Truncate(displas_vidsutni_vidom_skr2_cinI_step1 * cinI_zd);
displas_nayavni_vidom_skr2_cinI_step2 = Math.Truncate(displas_nayavni_vidom_skr2_cinI_step1 * cinI_cinI);
displas_nayavni_vidom_skr2_cinII_step2 = Math.Truncate(displas_nayavni_vidom_skr2_cinI_step1 * cinI_cinII) +
Math.Truncate(displas_nayavni_vidom_skr2_cinII_step1 * cinII_cinII) + Math.Truncate(displas_vidsutni_vidom_skr2_cinI_step1 * cinI_cinII);
displas_nayavni_vidom_skr2_rshm_step2 = Math.Truncate(displas_nayavni_vidom_skr2_cinII_step1 * cinII_rshm);
displas_vidsutni_vidom_skr2_zd_step2 = Math.Truncate(displas_vidsutni_vidom_skr2_zd_step1 * zd_zd) +
Math.Truncate(displas_nayavni_vidom_skr2_zd_step1 * zd_zd);
displas_vidsutni_vidom_skr2_cinI_step2 = Math.Truncate(displas_nayavni_vidom_skr2_zd_step1 * zd_cinI) +
Math.Truncate(displas_vidsutni_vidom_skr2_cinI_step1 * cinI_cinI);
////sum and skr2
displas_vidsutni_vidom_step2 = displas_nayavni_vidsutni_vidom_step2 + displas_vidsutni_vidsutni_vidom_step2 +
displas_vidsutni_vidom_skr2_zalysh_step1; //for next step
displas_nayavni_vidom_step2 = displas_nayavni_nayavni_vidom_step2 + displas_vidsutni_nayavni_vidom_step2 +
displas_nayavni_vidom_skr2_zalysh_step1; //for next step

displas_vidsutni_vidom_skr2_step2 = Math.Truncate(displas_vidsutni_vidom_step1 * ver_skrining_vids);//go next
displas_vidsutni_vidom_skr2_zalysh_step2 = displas_vidsutni_vidom_step2 - displas_vidsutni_vidom_skr2_step2;
displas_nayavni_vidom_skr2_step2 = Math.Truncate(displas_nayavni_vidom_step1 * ver_skrining_nayavn);//go next
displas_nayavni_vidom_skr2_zalysh_step2 = displas_nayavni_vidom_step1 - displas_nayavni_vidom_skr2_step2;
////displas_nayavni_vidom_skr2_step1
vpl_prys_disp_nayavn_step2 = Math.Truncate(displas_nayavni_vidom_skr2_step2 * ver_vpl_prys_disp_nayavn);
vpl_vids_disp_nayavn_step2 = displas_nayavni_vidom_skr2_step2 - vpl_prys_disp_nayavn_step2;
////displas_vids_vidom_skr2_step1
vpl_vids_disp_nema_step2 = Math.Truncate(displas_vidsutni_vidom_skr2_step2 * ver_vpl_vids_disp_nema);
vpl_prys_disp_nema_step2 = displas_vidsutni_vidom_skr2_step2 - vpl_vids_disp_nema_step2;
 + Math.Truncate(vpl_prys_disp_nema_step2 * ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nema_step2 * ver_povtorn_skr_1); //vot tut
nujen counter vitraty
////stay going next step
vpl_prys_disp_nayavn_stay_step2 = vpl_prys_disp_nayavn_step2 - prohodyt_med_dopomog_vpl_prys_disp_nayavn_step2 -
Math.Truncate(vpl_prys_disp_nayavn_step2 * ver_povtorn_skr_1);
vpl_prys_disp_nema_stay_step2 = vpl_prys_disp_nema_step2 - Math.Truncate(vpl_prys_disp_nema_step2 * ver_povtorn_skr_1);
vpl_vids_disp_nayavn_stay_step2 = vpl_vids_disp_nayavn_step2 - prohodyt_med_dopomog_vpl_vids_disp_nayavn_step2 -
Math.Truncate(vpl_vids_disp_nayavn_step2 * ver_povtorn_skr_1);
```

```
            vpl_vids_disp_nema_stay_step2 = vpl_vids_disp_nema_step2 - Math.Truncate(vpl_vids_disp_nema_step2 * ver_povtorn_skr_1);
            vpl_prys_disp_nayavn_stay_step2_zd = Math.Truncate(vpl_prys_disp_nayavn_stay_step1_cinI * cinI_zd) +
Math.Truncate(vpl_prys_disp_nema_stay_step1_cinI * cinI_zd);
            vpl_prys_disp_nayavn_stay_step2_cinI = Math.Truncate(vpl_prys_disp_nayavn_stay_step1_zd * zd_cinI) +
Math.Truncate(vpl_prys_disp_nema_stay_step1_zd * zd_cinI) +
                Math.Truncate(vpl_prys_disp_nema_stay_step1_cinI * cinI_cinI) + Math.Truncate(vpl_prys_disp_nayavn_stay_step2_cinI * cinI_cinI);
            vpl_prys_disp_nayavn_stay_step2_cinII = Math.Truncate(vpl_prys_disp_nayavn_stay_step1_cinI * cinI_cinII) +
Math.Truncate(vpl_prys_disp_nema_stay_step1_cinI * cinI_cinII);
            vpl_prys_disp_nayavn_stay_step2_rshm = Math.Truncate(vpl_prys_disp_nayavn_stay_step1_cinII * cinII_rshm);
            vpl_prys_disp_nema_stay_step2_zd = Math.Truncate(vpl_prys_disp_nema_stay_step1_zd * zd_zd_l_vp) +
Math.Truncate(vpl_prys_disp_nayavn_stay_step1_cinI * zd_cinI_l_vp)
                + Math.Truncate(vpl_prys_disp_nayavn_stay_step1_zd * zd_zd_l_vp) + Math.Truncate(vpl_prys_disp_nema_stay_step1_cinI * cinI_zd_l_vp);
            vpl_prys_disp_nema_stay_step2_cinI = Math.Truncate(vpl_prys_disp_nema_stay_step1_cinI * cinI_cinI_l_vp) +
Math.Truncate(vpl_prys_disp_nayavn_stay_step1_zd * zd_cinI_l_vp);
            vpl_vids_disp_nayavn_stay_step2_zd = Math.Truncate(vpl_vids_disp_nayavn_stay_step1_cinI * cinI_zd_l_vv) +
Math.Truncate(vpl_vids_disp_nema_stay_step1_cinI * cinI_zd_l_vv);
            vpl_vids_disp_nayavn_stay_step2_cinI = Math.Truncate(vpl_vids_disp_nayavn_stay_step1_zd * zd_cinI_l_vv) +
Math.Truncate(vpl_vids_disp_nema_stay_step1_zd * zd_cinI_l_vv) +
                Math.Truncate(vpl_vids_disp_nema_stay_step1_cinI * cinI_cinI_l_vv) + Math.Truncate(vpl_vids_disp_nayavn_stay_step2_cinI *
cinI_cinI_l_vv);
            vpl_vids_disp_nayavn_stay_step2_cinII = Math.Truncate(vpl_vids_disp_nayavn_stay_step1_cinI * cinI_cinII_l_vv) +
Math.Truncate(vpl_vids_disp_nema_stay_step1_cinI * cinI_cinII_l_vv);
            vpl_vids_disp_nayavn_stay_step2_rshm = Math.Truncate(vpl_vids_disp_nayavn_stay_step1_cinII * cinII_rshm_l_vv);
            vpl_vids_disp_nema_stay_step2_zd = Math.Truncate(vpl_vids_disp_nema_stay_step1_zd * zd_zd_bl_vv) +
Math.Truncate(vpl_vids_disp_nayavn_stay_step1_cinI * cinI_zd_bl_vv)
                + Math.Truncate(vpl_vids_disp_nayavn_stay_step1_zd * zd_zd_bl_vv) + Math.Truncate(vpl_vids_disp_nema_stay_step1_cinI * cinI_zd_bl_vv);
            vpl_vids_disp_nema_stay_step2_cinI = Math.Truncate(vpl_vids_disp_nema_stay_step1_cinI * cinI_cinI_bl_vv) +
Math.Truncate(vpl_vids_disp_nayavn_stay_step1_zd * zd_cinI_bl_vv);
           double[] yValues1 = { (displas_nayavni_nevidom_zd_step2 + displas_vidsutni_nevidom_zd_step2 + displas_nayavni_vidom_skr2_zd_step2 +
                    displas_vidsutni_vidom_skr2_zd_step2 + vpl_prys_disp_nayavn_stay_step2_zd + vpl_prys_disp_nema_stay_step2_zd +
                    vpl_vids_disp_nayavn_stay_step2_zd + vpl_vids_disp_nema_stay_step1_zd),
            (displas_nayavni_nevidom_cinI_step2 + displas_vidsutni_nevidom_cinI_step2 + displas_nayavni_vidom_skr2_cinI_step2
                    + displas_vidsutni_vidom_skr2_cinI_step2 + vpl_prys_disp_nayavn_stay_step2_cinI + vpl_prys_disp_nema_stay_step2_cinI
                    + vpl_vids_disp_nayavn_stay_step2_cinI + vpl_vids_disp_nema_stay_step1_cinI),
            (displas_nayavni_nevidom_cinII_step2 + displas_vidsutni_nevidom_cinII_step2 + displas_nayavni_vidom_skr2_cinII_step2
                      + vpl_prys_disp_nayavn_stay_step2_cinII + vpl_vids_disp_nayavn_stay_step2_cinII),
            (displas_nayavni_nevidom_rshm_step2 + displas_nayavni_vidom_skr2_rshm_step2 + vpl_prys_disp_nayavn_stay_step2_rshm
                      + vpl_vids_disp_nayavn_stay_step2_rshm), 0, (displas_nayavni_nevidom_zalysh_step2 + displas_vidsutni_nevidom_zalysh_step2)};
                  string[] xValues1 = { "", "", "", "", "", "" };
            for (int s2 = 0; s2 < 38; s2++)
            {
              dataGridView10.Rows[s2].Cells[1].Value = Math.Truncate((displas_nayavni_nevidom_zd_step2 + displas_vidsutni_nevidom_zd_step2 +
displas_nayavni_vidom_skr2_zd_step2 +
                    displas_vidsutni_vidom_skr2_zd_step2 + vpl_prys_disp_nayavn_stay_step2_zd + vpl_prys_disp_nema_stay_step2_zd +
                    vpl_vids_disp_nayavn_stay_step2_zd + vpl_vids_disp_nema_stay_step1_zd) * age_part[s2] * best_population_part[s2, 0]);
              dataGridView10.Rows[s2].Cells[2].Value = Math.Truncate((displas_nayavni_nevidom_cinI_step2 + displas_vidsutni_nevidom_cinI_step2 +
displas_nayavni_vidom_skr2_cinI_step2
                    + displas_vidsutni_vidom_skr2_cinI_step2 + vpl_prys_disp_nayavn_stay_step2_cinI + vpl_prys_disp_nema_stay_step2_cinI
                    + vpl_vids_disp_nayavn_stay_step2_cinI + vpl_vids_disp_nema_stay_step1_cinI) * age_part[s2] * best_population_part[s2, 1]);
              dataGridView10.Rows[s2].Cells[3].Value = Math.Truncate((displas_nayavni_nevidom_cinII_step2 + displas_vidsutni_nevidom_cinII_step2 +
displas_nayavni_vidom_skr2_cinII_step2
                      + vpl_prys_disp_nayavn_stay_step2_cinII + vpl_vids_disp_nayavn_stay_step2_cinII) * age_part[s2] * best_population_part[s2, 2]);
```

```
            dataGridView10.Rows[s2].Cells[4].Value = Math.Truncate((displas_nayavni_nevidom_rshm_step2 + displas_nayavni_vidom_skr2_rshm_step2 +
vpl_prys_disp_nayavn_stay_step2_rshm
                        + vpl_vids_disp_nayavn_stay_step2_rshm) * age_part[s2] * best_population_part[s2, 3]);
        }
        ////displas_nayavni_nevidom
        displas_nayavni_nevidom_skr1_step3 = Math.Truncate(displas_nayavni_nevidom_zalysh_step2 * ver_skrining1) +
Math.Truncate((Math.Truncate(vpl_prys_disp_nayavn_step2 * ver_povtorn_skr_1)
            + Math.Truncate(vpl_vids_disp_nayavn_step2 * ver_povtorn_skr_1)) * ver_skrining1);
        displas_nayavni_nevidom_zalysh_step3 = displas_nayavni_nevidom_zalysh_step2 + Math.Truncate(vpl_prys_disp_nayavn_step2 *
ver_povtorn_skr_1) + Math.Truncate(vpl_vids_disp_nayavn_step2 * ver_povtorn_skr_1) - displas_nayavni_nevidom_skr1_step3;
        displas_nayavni_nevidom_zd_step3 = Math.Truncate(displas_nayavni_nevidom_zd_step2 * zd_zd) +
Math.Truncate(displas_nayavni_nevidom_cinI_step2 * cinI_zd);
        displas_nayavni_nevidom_cinI_step3 = Math.Truncate(displas_nayavni_nevidom_cinI_step2 * cinI_cinI) +
Math.Truncate(displas_nayavni_nevidom_zd_step2 * zd_cinI);
        displas_nayavni_nevidom_cinII_step3 = Math.Truncate(displas_nayavni_nevidom_cinII_step2 * cinII_cinII) +
Math.Truncate(displas_nayavni_nevidom_cinI_step2 * cinI_cinII);
        displas_nayavni_nevidom_rshm_step3 = Math.Truncate(displas_nayavni_nevidom_cinII_step2 * cinII_rshm);
        ////displas_vidsutni_nevidom
        displas_vidsutni_nevidom_skr1_step3 = Math.Truncate(displas_vidsutni_nevidom_zalysh_step2 * ver_skrining1) +
Math.Truncate((Math.Truncate(vpl_prys_disp_nema_step2 * ver_povtorn_skr_1)
            + Math.Truncate(vpl_vids_disp_nema_step2 * ver_povtorn_skr_1)) * ver_skrining1);
        displas_vidsutni_nevidom_zalysh_step3 = displas_vidsutni_nevidom_zalysh_step2 + Math.Truncate(vpl_prys_disp_nema_step2 *
ver_povtorn_skr_1)
            + Math.Truncate(vpl_vids_disp_nema_step2 * ver_povtorn_skr_1) - displas_vidsutni_nevidom_skr1_step3;
        displas_vidsutni_nevidom_zd_step3 = Math.Truncate(displas_vidsutni_nevidom_zd_step2 * zd_zd) +
Math.Truncate(displas_vidsutni_nevidom_cinI_step2 * cinI_zd);
        displas_vidsutni_nevidom_cinI_step3 = Math.Truncate(displas_vidsutni_nevidom_cinI_step2 * cinI_cinI) +
Math.Truncate(displas_vidsutni_nevidom_zd_step2 * zd_cinI);
        displas_vidsutni_nevidom_cinII_step3 = Math.Truncate(displas_vidsutni_nevidom_cinI_step2 * cinII_cinII);
        ////displas_nayavni_nevidom skr 1
        displas_nayavni_nayavni_vidom_step3 = Math.Truncate(displas_nayavni_nevidom_skr1_step3 * zd_cinI); //nayavni
        displas_nayavni_vidsutni_vidom_step3 = displas_nayavni_nevidom_skr1_step3 - displas_nayavni_nayavni_vidom_step3; //vidsutni
        ////displas_vidsutni_nevidom skr 1
        displas_vidsutni_nayavni_vidom_step3 += Math.Truncate(displas_vidsutni_nevidom_skr1_step3 * zd_cinI); //nayavni
        displas_vidsutni_vidsutni_vidom_step3 = displas_vidsutni_nevidom_skr1_step3 - displas_vidsutni_nayavni_vidom_step3; //vidsutni
        ////neproshli skrin 2 _ step 0 perehodi vnutri po step 0
        displas_nayavni_vidom_skr2_zd_step3 = Math.Truncate(displas_nayavni_vidom_skr2_cinI_step2 * cinI_zd) +
Math.Truncate(displas_vidsutni_vidom_skr2_cinI_step2 * cinI_zd);
        displas_nayavni_vidom_skr2_cinI_step3 = Math.Truncate(displas_nayavni_vidom_skr2_cinI_step2 * cinI_cinI);
        displas_nayavni_vidom_skr2_cinII_step3 = Math.Truncate(displas_nayavni_vidom_skr2_cinI_step2 * cinI_cinII) +
Math.Truncate(displas_nayavni_vidom_skr2_cinII_step2 * cinII_cinII) + Math.Truncate(displas_vidsutni_vidom_skr2_cinI_step2 * cinI_cinII);
        displas_nayavni_vidom_skr2_rshm_step3 = Math.Truncate(displas_nayavni_vidom_skr2_cinII_step2 * cinII_rshm);
        displas_vidsutni_vidom_skr2_zd_step3 = Math.Truncate(displas_vidsutni_vidom_skr2_zd_step2 * zd_zd) +
Math.Truncate(displas_nayavni_vidom_skr2_zd_step2 * zd_zd);
        displas_vidsutni_vidom_skr2_cinI_step3 = Math.Truncate(displas_nayavni_vidom_skr2_zd_step2 * zd_cinI) +
Math.Truncate(displas_vidsutni_vidom_skr2_cinI_step2 * cinI_cinI);
        ////sum and skr2
        displas_vidsutni_vidom_step3 = displas_nayavni_vidsutni_vidom_step3 + displas_vidsutni_vidsutni_vidom_step3 +
displas_vidsutni_vidom_skr2_zalysh_step2; //for next step
        displas_nayavni_vidom_step3 = displas_nayavni_nayavni_vidom_step3 + displas_vidsutni_nayavni_vidom_step3 +
displas_nayavni_vidom_skr2_zalysh_step2; //for next step
        displas_vidsutni_vidom_skr2_step3 = Math.Truncate(displas_vidsutni_vidom_step2 * ver_skrining_vids);//go next
```

```
displas_vidsutni_vidom_skr2_zalysh_step3 = displas_vidsutni_vidom_step3 - displas_vidsutni_vidom_skr2_step3;
displas_nayavni_vidom_skr2_step3 = Math.Truncate(displas_nayavni_vidom_step2 * ver_skrining_nayavn);//go next
displas_nayavni_vidom_skr2_zalysh_step3 = displas_nayavni_vidom_step2 - displas_nayavni_vidom_skr2_step3;
////displas_nayavni_vidom_skr2_step2
vpl_prys_disp_nayavn_step3 = Math.Truncate(displas_nayavni_vidom_skr2_step3 * ver_vpl_prys_disp_nayavn);
vpl_vids_disp_nayavn_step3 = displas_nayavni_vidom_skr2_step3 - vpl_prys_disp_nayavn_step3;
////displas_vids_vidom_skr2_step2
vpl_vids_disp_nema_step3 = Math.Truncate(displas_vidsutni_vidom_skr2_step3 * ver_vpl_vids_disp_nema);
vpl_prys_disp_nema_step3 = displas_vidsutni_vidom_skr2_step3 - vpl_vids_disp_nema_step3;
 ////stay going next step
vpl_prys_disp_nayavn_stay_step3 = vpl_prys_disp_nayavn_step3 - prohodyt_med_dopomog_vpl_prys_disp_nayavn_step3 -
Math.Truncate(vpl_prys_disp_nayavn_step3 * ver_povtorn_skr_1);
vpl_prys_disp_nema_stay_step3 = vpl_prys_disp_nema_step3 - Math.Truncate(vpl_prys_disp_nema_step3 * ver_povtorn_skr_1);
vpl_vids_disp_nayavn_stay_step3 = vpl_vids_disp_nayavn_step3 - prohodyt_med_dopomog_vpl_vids_disp_nayavn_step3 -
Math.Truncate(vpl_vids_disp_nayavn_step3 * ver_povtorn_skr_1);
vpl_vids_disp_nema_stay_step3 = vpl_vids_disp_nema_step3 - Math.Truncate(vpl_vids_disp_nema_step3 * ver_povtorn_skr_1);
vpl_prys_disp_nayavn_stay_step3_zd = Math.Truncate(vpl_prys_disp_nayavn_stay_step2_cinI * cinI_zd) +
Math.Truncate(vpl_prys_disp_nema_stay_step2_cinI * cinI_zd);
vpl_prys_disp_nayavn_stay_step3_cinI = Math.Truncate(vpl_prys_disp_nayavn_stay_step2_zd * zd_cinI) +
Math.Truncate(vpl_prys_disp_nema_stay_step2_zd * zd_cinI) +
    Math.Truncate(vpl_prys_disp_nema_stay_step2_cinI * cinI_cinI) + Math.Truncate(vpl_prys_disp_nayavn_stay_step3_cinI * cinI_cinI);
vpl_prys_disp_nayavn_stay_step3_cinII = Math.Truncate(vpl_prys_disp_nayavn_stay_step2_cinI * cinI_cinII) +
Math.Truncate(vpl_prys_disp_nema_stay_step2_cinI * cinI_cinII);
vpl_prys_disp_nayavn_stay_step3_rshm = Math.Truncate(vpl_prys_disp_nayavn_stay_step2_cinII * cinII_rshm) +
Math.Truncate(vpl_prys_disp_nayavn_stay_step2_zd * zd_cinI);
vpl_prys_disp_nayavn_stay_step3_smrt = Math.Truncate(vpl_prys_disp_nayavn_stay_step2_rshm * rshm_smert);
//vpl_prys_disp_nema_stay_step2_zd <- vozmojno stoit brat' takie znacheniya, a ne turncate
vpl_prys_disp_nema_stay_step3_zd = Math.Truncate(vpl_prys_disp_nema_stay_step2_zd * zd_zd_l_vp) +
Math.Truncate(vpl_prys_disp_nayavn_stay_step2_cinI * cinI_zd_l_vp)
    + Math.Truncate(vpl_prys_disp_nayavn_stay_step2_zd * zd_zd_l_vp) + Math.Truncate(vpl_prys_disp_nema_stay_step2_cinI * cinI_zd_l_vp);
vpl_prys_disp_nema_stay_step3_cinI = Math.Truncate(vpl_prys_disp_nema_stay_step2_cinI * cinI_cinI_l_vp) +
Math.Truncate(vpl_prys_disp_nayavn_stay_step2_zd * zd_cinI_l_vp);
vpl_vids_disp_nayavn_stay_step3_zd = Math.Truncate(vpl_vids_disp_nayavn_stay_step2_cinI * cinI_zd_l_vv) +
Math.Truncate(vpl_vids_disp_nema_stay_step2_cinI * cinI_zd_l_vv);
vpl_vids_disp_nayavn_stay_step3_cinI = Math.Truncate(vpl_vids_disp_nayavn_stay_step2_zd * zd_cinI_l_vv) +
Math.Truncate(vpl_vids_disp_nema_stay_step2_zd * zd_cinI_l_vv) +
    Math.Truncate(vpl_vids_disp_nema_stay_step2_cinI * cinI_cinI_l_vv) + Math.Truncate(vpl_vids_disp_nayavn_stay_step3_cinI *
cinI_cinI_l_vv);
vpl_vids_disp_nayavn_stay_step3_cinII = Math.Truncate(vpl_vids_disp_nayavn_stay_step2_cinI * cinI_cinII_l_vv) +
Math.Truncate(vpl_vids_disp_nema_stay_step2_cinI * cinI_cinII_l_vv);
vpl_vids_disp_nayavn_stay_step3_rshm = Math.Truncate(vpl_vids_disp_nayavn_stay_step2_cinII * cinII_rshm_l_vv) +
Math.Truncate(vpl_vids_disp_nayavn_stay_step2_rshm * rshm_rshm_l_vv);
vpl_vids_disp_nayavn_stay_step3_smrt = Math.Truncate(vpl_vids_disp_nayavn_stay_step2_rshm * rshm_smert_l_vv);
vpl_vids_disp_nema_stay_step3_zd = Math.Truncate(vpl_vids_disp_nema_stay_step2_zd * zd_zd_bl_vv) +
Math.Truncate(vpl_vids_disp_nayavn_stay_step2_cinI * cinI_zd_bl_vv)
    + Math.Truncate(vpl_vids_disp_nayavn_stay_step2_zd * zd_zd_bl_vv) + Math.Truncate(vpl_vids_disp_nema_stay_step2_cinI * cinI_zd_bl_vv);
vpl_vids_disp_nema_stay_step3_cinI = Math.Truncate(vpl_vids_disp_nema_stay_step2_cinI * cinI_cinI_bl_vv) +
Math.Truncate(vpl_vids_disp_nayavn_stay_step2_zd * zd_cinI_bl_vv);
double[] yValues2 = { (displas_nayavni_nevidom_zd_step3 + displas_vidsutni_nevidom_zd_step3 + displas_nayavni_vidom_skr2_zd_step3 +
        displas_vidsutni_vidom_skr2_zd_step3 + vpl_prys_disp_nayavn_stay_step3_zd + vpl_prys_disp_nema_stay_step3_zd +
        vpl_vids_disp_nayavn_stay_step3_zd + vpl_vids_disp_nema_stay_step1_zd),
(displas_nayavni_nevidom_cinI_step3 + displas_vidsutni_nevidom_cinI_step3 + displas_nayavni_vidom_skr2_cinI_step3
```

```
                + displas_vidsutni_vidom_skr2_cinI_step3 + vpl_prys_disp_nayavn_stay_step3_cinI + vpl_prys_disp_nema_stay_step3_cinI
                + vpl_vids_disp_nayavn_stay_step3_cinI + vpl_vids_disp_nema_stay_step1_cinI),
        (displas_nayavni_nevidom_cinII_step3 + displas_vidsutni_nevidom_cinII_step3 + displas_nayavni_vidom_skr2_cinII_step3
                + vpl_prys_disp_nayavn_stay_step3_cinII + vpl_vids_disp_nayavn_stay_step3_cinII),
        (displas_nayavni_nevidom_rshm_step3 + displas_nayavni_vidom_skr2_rshm_step3 + vpl_prys_disp_nayavn_stay_step3_rshm
                + vpl_vids_disp_nayavn_stay_step3_rshm), (vpl_vids_disp_nayavn_stay_step3_smrt + vpl_prys_disp_nayavn_stay_step3_smrt),
        (displas_nayavni_nevidom_zalysh_step3 + displas_vidsutni_nevidom_zalysh_step3)};
        string[] xValues2 = { "", "", "", "", "", "" };
        for (int s3 = 0; s3 < 38; s3++)
        {
          dataGridView11.Rows[s3].Cells[1].Value = Math.Truncate((displas_nayavni_nevidom_zd_step3 + displas_vidsutni_nevidom_zd_step3 +
displas_nayavni_vidom_skr2_zd_step3 +
                displas_vidsutni_vidom_skr2_zd_step3 + vpl_prys_disp_nayavn_stay_step3_zd + vpl_prys_disp_nema_stay_step3_zd +
                vpl_vids_disp_nayavn_stay_step3_zd + vpl_vids_disp_nema_stay_step1_zd) * age_part[s3] * best_population_part[s3, 0]);
          dataGridView11.Rows[s3].Cells[2].Value = Math.Truncate((displas_nayavni_nevidom_cinI_step3 + displas_vidsutni_nevidom_cinI_step3 +
displas_nayavni_vidom_skr2_cinI_step3
                + displas_vidsutni_vidom_skr2_cinI_step3 + vpl_prys_disp_nayavn_stay_step3_cinI + vpl_prys_disp_nema_stay_step3_cinI
                + vpl_vids_disp_nayavn_stay_step3_cinI + vpl_vids_disp_nema_stay_step1_cinI) * age_part[s3] * best_population_part[s3, 1]);
          dataGridView11.Rows[s3].Cells[3].Value = Math.Truncate((displas_nayavni_nevidom_cinII_step3 + displas_vidsutni_nevidom_cinII_step3 +
displas_nayavni_vidom_skr2_cinII_step3
                + vpl_prys_disp_nayavn_stay_step3_cinII + vpl_vids_disp_nayavn_stay_step3_cinII) * age_part[s3] * best_population_part[s3, 2]);
          dataGridView11.Rows[s3].Cells[4].Value = Math.Truncate((displas_nayavni_nevidom_rshm_step3 + displas_nayavni_vidom_skr2_rshm_step3 +
vpl_prys_disp_nayavn_stay_step3_rshm
                + vpl_vids_disp_nayavn_stay_step3_rshm) * age_part[s3] * best_population_part[s3, 3]);
          dataGridView11.Rows[s3].Cells[5].Value = Math.Truncate((vpl_vids_disp_nayavn_stay_step3_smrt + vpl_prys_disp_nayavn_stay_step3_smrt) *
age_part[s3] * best_population_part[s3, 4]);
          values_for_comp_chart += Convert.ToInt32(dataGridView11.Rows[s3].Cells[3].Value) +
Convert.ToInt32(dataGridView11.Rows[s3].Cells[4].Value);
          values_for_comp_chart_exit += Convert.ToInt32(Convert.ToDouble(dataGridView11.Rows[s3].Cells[3].Value) *
Convert.ToDouble(dataGridView11.Rows[s3].Cells[4].Value) * matr_rand_3[2] + matr_rand_4[3]);
          chart3.Series[0].Points.AddXY(population[s3, 0], Convert.ToInt32(dataGridView11.Rows[s3].Cells[3].Value) +
Convert.ToInt32(dataGridView11.Rows[s3].Cells[3].Value));
        }
```

ДОДАТОК В

Інформаційна технологія для ефективного менеджменту хронічних захворювань

Опис програмного коду

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5274_19Б

Аркушів 8

Київ 2019

АНОТАЦІЯ

Даний додаток містить опис інформаційної системи розробленої для ефективного менеджменту хронічних захворювань. Створений програмний продукт демонструє розвиток двох популяцій, контрольної та оптимальної та виконує такі завдання:

– моделювання контрольної та оптимальної популяції людей, з урахуванням прогресії захворювання;

– порівняння отриманих груп людей і оцінка ефективності;

– отримання оптимальної стратегії медичного втручання у хід розвитку хвороби;

– виведення графіків залежності;

– можливість генерувати популяцію за вже існуючим сценарієм фармакотерапії.

Програма отримує дані через елементи керування, які розташовані на Windows Form.

При розробці програмної системи була використана об’єктно-орієнтовна мова програмування C# та середовище розробки Microsoft Visual Studio 2017 з використанням графічного інтерфейсу Windows Forms та відповідних бібліотек.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис програмної інформаційної системи для ефективного менеджменту хронічних захворювань. У додатку Б міститься програмний код головних модулів розроблюваної програмної системи

Розроблений додаток працює в операційній системі Windows Vista, Windows XP, Windows 7 та Windows 10 і потребує встановленого пакету Microsoft Office на комп’ютері.

При розробці програмного продукту використовувалась об’єктно-орієнтовна мова програмування C# та середовище розробки Microsoft Visual Studio 2017.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблені компоненти виконують завдання моделювання популяцій з хронічними хворобами і різною стратегією фармакотерапії та отримання найбільш ефективної стратегії. Це відбувається за допомогою методів імітаційного моделювання, методу Монте-Карло і марковського процесу прийняття рішень.

Розроблений додаток може використовуватися на база медичних закладів та установ в якості аналітичної програми.

Функціональні обмеження на використання компонентів полягають лише в розмірі вхідної популяції.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Для забезпечення повноцінної роботи та досягнення високої точності роботи створених додатків для ефективного менеджменту хронічних захворювань було обрано середовище розробки Microsoft Visual Studio 2017, яке показало себе як надійне, зручне та гнучке для розробника рішення.

Також використовувалися додаткові бібліотеки Microsoft.Office.Interop.Excel для роботи з документами програми Microsoft Office Excel та System.Drawing – для роботи з графічним інтерфейсом.

Розроблений додаток працює під операційними системами Windows XP, Windows Vista, Windows 7 та Windows 10 і потребує встановленого на комп’ютері пакету програм Microsoft Office.

4 ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для реалізації системи для ефективного менеджменту хронічних захворювань знадобилася розробка алгоритму.

Він будується на основі методів імітаційного моделювання, методів Монте-Карло і марковського процесу прийняття рішень.

Загалом, моделювання популяцій людей відбувається відповідно методам імітаційного моделювання. Марковський процес прийняття рішень потрібен у системі, щоб моделювати розвиток хвороби у осіб популяції та робити відповідні переходи. Основа методу Монте-Карло використовується для прискорення роботи програмного продукту та збільшення його точності.

Під час запуску програмного додатку, з’являється вікно головної форми програми, де користувач повинен загрузити необхідні вхідні дані та ввести початкові параметри. Після чого данні передаються до програми, обчислюються, а кінцевий результат повертається у форму або зберігається користувачем на комп’ютер.

5 ВИКЛИК І ЗАВАНТАЖЕНЯ

Розроблена система не потребує додаткової інсталяції. Для того, щоб працювати з системою потрібно лише запустити виконуваний файл “WindowsFormsApp1.exe”.

Після запуску користувач переходить до головної форми програмного продукту, де може проводити необхідні дії.

6 ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для розробленого додатку є файл у форматі .xls або .xlsx з даними про певну популяційну вибірку, а також розмір популяції, який він може встановити у відповідному textBox.

Вихідними даними є набір графіків залежностей, які користувач може зберегти собі на комп’ютер як окремо, так і у комплексному звіті у форматі документа Excel.

Вихідними даними також є набір популяцій розбитих та структурованих за певними характеристиками і параметрами.